



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دستور کار آزمایشگاه هوش محاسباتی
دانشگاه مهندسی برق

تهریه کنندگان : پریسا داج خوش , کوثر بهنیا , علی محمد حسینی , بهنام
جبار زاده , محمد حسین امینی

مسئول آزمایشگاه : دکتر فرزانه عبدالهی

فهرست مطالب

۱۰	چکیده	۶
آزمایش اول (آشنایی با متلب و پایتون)		
۱.۱	پیش گزارش	۸
۲.۱	مقدمه	۸
۱.۲.۱	آشنایی با کد نویس در متلب	۸
۲.۲.۱	آشنایی با کد نویسی در پایتون	۱۰
۳.۱	شرح آزمایش	۱۱
۴.۱	تمرین	۱۲
آزمایش دوم(پیاده سازی پرسپترون)		
۱.۲	پیش گزارش	۱۴
۲.۲	مقدمه	۱۴
۱.۲.۲	الگوریتم یادگیری	۱۵
۳.۲	شرح آزمایش	۱۵
۱.۳.۲	پیاده سازی پرسپترون	۱۶
۲.۳.۲	داده ورودی	۱۶
۳.۳.۲	بهبود مدل	۱۶
۴.۲	تمرین	۱۶
آزمایش سوم(پیاده سازی شبکه عصبی چند لایه)		
۱.۳	پیش گزارش	۱۹
۲.۳	مقدمه	۱۹
۱.۲.۳	الگوریتم پس انتشار خط	۲۰

۲۱	۳.۳	شرح آزمایش
۲۱	۱.۳.۳	پیاده سازی شبکه عصبی الایه
۲۱	۲.۳.۳	تست مدل
۲۲	۴.۳	تمرین
۲۳	(<i>k - mean</i> الگوریتم)	آزمایش چهارم (پیاده سازی الگوریتم <i>k - mean</i>)	
۲۴	۱.۴	پیش گزارش
۲۴	۲.۴	مقدمه
۲۴	۱.۲.۴	الگوریتم <i>k - mean</i>
۲۵	۳.۴	شرح آزمایش
۲۵	۱.۳.۴	پیاده سازی الگوریتم
۲۶	۲.۳.۴	تست الگوریتم
۲۶	۳.۳.۴	ارزیابی الگوریتم
۲۷	۴.۳.۴	محدودیت های <i>kmean</i>
۲۸	۴.۴	کاهش حجم عکس به وسیله <i>k - mean</i>
۲۸	۵.۴	تمرین
۲۹	(<i>RBF</i> پیاده سازی شبکه عصبی (<i>RBF</i>))	آزمایش پنجم(پیاده سازی شبکه عصبی (<i>RBF</i>))	
۳۰	۱.۵	پیش گزارش
۳۰	۲.۵	مقدمه
۳۰	۱.۲.۵	ساختار شبکه
۳۱	۲.۲.۵	تابع گوسین
۳۱	۳.۲.۵	تابع گوسین و <i>RBF</i>
۳۲	۴.۲.۵	قانون تازه سازی وزن های شبکه
۳۳	۳.۵	شرح آزمایش
۳۳	۱.۳.۵	پیاده سازی شبکه
۳۴	۲.۳.۵	تست شبکه
۳۴	۴.۵	تمرین
۳۵	(<i>Hopfield</i> پیاده سازی شبکه عصبی (<i>Hopfield</i>))	آزمایش ششم (پیاده سازی شبکه عصبی (<i>Hopfield</i>))	
۳۶	۱.۶	پیش گزارش

۳۶	۲.۶	مقدمه
۳۷	۱.۲.۶	تحلیل فیزیکی حافظه
۳۷	۲.۲.۶	شبکه هاپفیلد
۳۸	۳.۶	شرح آزمایش
۳۸	۴.۶	تمرین
۴۰		آزمایش هفتم (شناسایی به کمک شبکه عصبی)
۴۱	۱.۷	پیش گزارش
۴۱	۲.۷	مقدمه
۴۱	۱.۲.۷	شناساگر استاتیکی
۴۱	۲.۲.۷	شناساگر پیوسته
۴۲	۳.۲.۷	مدل موازی
۴۳	۴.۲.۷	ترکیب شناساگر استاتیکی و مدل موازی
۴۴	۳.۷	شرح آزمایش
۴۴	۱.۳.۷	پیاده سازی شناساگر استاتیکی
۴۶	۲.۳.۷	پیاده سازی شناساگر پیوسته
۴۸	۳.۳.۷	تست شناساگر شبیه سازی شده
۴۹	۴.۷	تمرین
۵۰		آزمایش هشتم (کنترل به کمک شبکه عصبی)
۵۱	۱.۸	پیش گزارش
۵۱	۲.۸	مقدمه
۵۱	۱.۲.۸	کنترل دینامیک معکوس
۵۲	۳.۸	شرح آزمایش
۵۲	۴.۸	تمرین
۵۳		پروژه بخش عصبی
۵۴	۱.۹	پیش گزارش
۵۴	۲.۹	مقدمه
۵۴	۱.۲.۹	خودرو زمینی بدون سرنشین
۵۵	۲.۲.۹	ربات نقاش ۲ بعدی

۵۵	۳.۹	شرح آزمایش
۵۵	۱.۳.۹	آزمایش ۱ - شناسایی خودرو زمینی بدون سرنشین
۵۶	۲.۳.۹	آزمایش ۲ - شناسایی ربات نقاش ۲ بعدی
۵۶	۳.۳.۹	آزمایش ۳ - کنترل خودروی زمینی بدون سرنشین
۵۷	۴.۹	تمرین
۵۸		آزمایش نهم (آشنایی با مفاهیم فازی)
۵۹	۱.۱	پیش گزارش
۵۹	۲.۱	مقدمه
۵۹	۱.۲.۱۰	تابع عضویت
۵۹	۲.۲.۱۰	نقطه عبور
۵۹	۳.۲.۱۰	عملگرها در منطق فازی
۶۰	۴.۲.۱۰	اجتماع
۶۰	۳.۱	شرح آزمایش
۶۱	۴.۱	تمرین
۶۳		آزمایش دهم(کنترلر فازی)
۶۴	۱.۱۱	پیش گزارش
۶۴	۲.۱۱	مقدمه
۶۴	۱.۲.۱۱	جعبه ابزار فازی متلب
۶۷	۳.۱۱	شرح آزمایش
۶۸	۴.۱۱	تمرین
۶۹		آزمایش یازدهم(کنترلر PID فازی)
۷۰	۱.۱۲	پیش گزارش
۷۰	۲.۱۲	مقدمه
۷۰	۱.۲.۱۲	کنترلر PID
۷۳	۳.۱۲	شرح آزمایش
۷۴	۴.۱۲	تمرین
۷۵		آزمایش دوازدهم($c - mean$)
۷۵	۱.۱۳	پیش گزارش

۷۵	۲.۱۳ مقدمه
۷۶	۱.۲.۱۳ آشنایی با جعبه ابزار <i>c – mean</i> فازی متلب
۷۶	۲.۲.۱۳ ابزار فازی در محیط <i>command – line</i> متلب
۷۷	۳.۱۳ شرح آزمایش
۷۷	۱.۳.۱۳ دسته بندی با استفاده از جعبه ابزار <i>fcm</i>
۷۷	۲.۳.۱۳ دسته بندی <i>fcm</i> در <i>Command – line</i>
۷۸	۴.۱۳ تمرین
۷۹	آزمایش سیزدهم (<i>anfis</i>)
۸۰	۱.۱۴ پیش گزارش
۸۰	۲.۱۴ مقدمه
۸۰	۱.۲.۱۴ جعبه ابزار <i>anfis</i> متلب
۸۱	۳.۱۴ شرح آزمایش
۸۱	۴.۱۴ تمرین
۸۲	پروژه بخش فازی
۸۳	۱.۱۵ پیش گزارش
۸۳	۲.۱۵ مقدمه
۸۳	۱.۲.۱۵ خودرو زمینی بدون سرنشین
۸۴	۲.۲.۱۵ ربات نقاش ۲ بعدی
۸۴	۳.۱۵ شرح آزمایش
۸۴	۱.۳.۱۵ آزمایش ۱ - کنترل خودروی زمینی بدون سرنشین
۸۵	۲.۳.۱۵ آزمایش ۲ - شناسایی ربات نقاش ۲ بعدی
۸۶	پیوست
۸۶	آ ربات نقاش ۲ بعدی
۸۸	آ.۱ ترسیم با ربات های نقاش
۸۹	ب روبات زمینی ((<i>UGV</i>))
۸۹	ب.۱ مدلسازی
۹۴	ب.۲ ارتباط با <i>UGV</i>

۱۰. چکیده

در این آزمایشگاه به پیاده سازی نرم افزاری و سخت افزاری مفاهیم بیان شده در درس هوش محاسباتی پرداخته می شود. گروه های در نظر گرفته شده برای این آزمایشگاه، گروه های ۳ نفره است. در ۸ آزمایش اول به پیاده سازی مفاهیم شبکه عصبی پرداخته می شود سپس در قالب یک پروژه به پیاده سازی سخت افزاری مفاهیم بیان شده پرداخته خواهد شد. در آزمایش نهم تا دوازدهم به پیاده سازی مفاهیم سیستم های فازی پرداخته می شود. در آزمایش ۱۳ با ترکیب سیستم های عصبی و فازی آشنا خواهید شد و سپس در قالب یک پروژه مفاهیم بیان شده بر روی سیستم های در نظر گرفته شده پیاده سازی سخت افزاری می شوند.

هر آزمایش از ۴ بخش پیش گزارش ، مقدمه ، شرح آزمایش و تمرین تشکیل شده است. پیش گزارش هر آزمایش را قبل از اجرای آزمایش به مسئول آزمایشگاه تحويل دهید. در قسمت مقدمه پیش نیازهای لازم برای هر آزمایش بیان شده است. در قسمت شرح آزمایش روند کلی و هدف هر آزمایش بیان شده است. در قسمت تمرین سوالاتی درباره آزمایش انجام شده در نظر گرفته شده است. تمرین هر آزمایش را قبل از شروع آزمایش بعد به مسئول مربوطه تحويل دهید(مسائل شبیه سازی را در moodle درس بار گزاری کنید).

آزمایش اول (آشنایی با متلب و پایتون)

۱.۱ پیش گزارش

۱. کاربردهای زبان برنامه نویسی پایتون را بیان کنید

۲. تفاوت زبان برنامه نویسی متلب و پایتون چیست؟

۳. مزیت های زبان پایتون در برابر متلب و بر عکس را بیان کنید.

۲.۱ مقدمه

در این آزمایش قصد داریم با زبان های برنامه نویسی متلب و پایتون آشنا شویم. در ادامه آزمایش ها در محیط متلب یا پایتون اجرا می شوند. در ابتدا مروری کوتاه بر روی کد نویسی در محیط های متلب و پایتون انجام می شود.

۱.۲.۱ آشنایی با کد نویس در متلب

متلب یکی از پر کاربرد ترین زبان های برنامه نویسی به شمار می رود. متلب دارای کتابخانه های آماده و مفید در حوزه های مختلفی مانند پردازش تصویر و هوش مصنوعی است. در این قسمت قصد داریم مروری کوتاه بر چگونگی برنامه نویسی در این محیط داشته باشیم. در ابتدا با چگونگی تعریف تابع در متلب آشنا خواهیم شد سپس مروری کوتاه بر روی آرایه ها در متلب انجام می شود با گذر از این موضوع به ترسیم نمودار در متلب پرداخته می شود. این بخش با معرفی کردن ابزار *help* در متلب پایان می پذیرد.

تعریف تابع

توابع در برنامه نویسی توسعه دادن کد را آسان تر می کنند. با کمک توابع توضیح کار کرد کد نیز ساده تر می شود. برای ایجاد یک تابع در متلب قدم های زیر را دنبال کنید:

۱. بر روی منوی *New* کلیک کرده و گزینه‌ی *Function* را انتخاب می کنیم

۲. در طرف چپ مساوی داخل کروشه خروجی‌های تابع با ذکر نام و تفکیک با ایجاد یک فاصله یا ویرگول، مشخص می‌شوند.

۳. در طرف راست مساوی نیز، داخل کروشه ورودی‌های تابع به صورت مشابه تعریف می‌شوند.

۴. در خط بعد روابط میان ورودی و خروجی‌ها تعریف می‌شود.

آرایه‌ها و ماتریس‌ها

آرایه‌ها و ماتریس‌ها کاربرد زیادی در انواع مسایل برنامه نویسی دارند. آرایه‌ها در مطلب به وسیله یک کروشه تعریف می‌شوند. اعضای آرایه با استفاده از ، و یا فاصله از یکدیگر جدا می‌شوند. شکل شماره ۱.۱ تعریف یک آرایه که دارای ۴ عضو است را در مطلب نمایش می‌دهد.

$$a = [1 \ 2 \ 3 \ 4]$$

شکل ۱.۱: تعریف آرایه در مطلب

ماتریس‌ها در مطلب همانند آرایه‌ها به وسیله کروشه مشخص می‌شوند سطر‌های یک ماتریس با استفاده از ”؛“ یا ”،“ از یکدیگر جدا می‌شوند.

شکل ۲.۱ چگونگی تعریف یک ماتریس در محیط مطلب را نمایش می‌دهد.

$$a = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 10]$$

شکل ۲.۱: تعریف ماتریس در مطلب

ترسیم نمودار

یکی از ویژگی‌های نرم افزار مطلب به تصویر کشیدن داده‌های مختلف و رسم نمودار است. مطلب امکان ترسیم نمودارهای ۲ بعدی و ۳ بعدی را فراهم می‌سازد.

برای رسم نمودار ۲ بعدی از دستور $plot(x, y)$ استفاده می‌شود. در این دستور x داده‌های متناظر بر روی محور افقی و y داده‌های متناظر بر روی محور عمودی را نمایش می‌دهد. دستور $plot$ ورودی‌های خود را به صورت آرایه دریافت می‌کند. برای رسم نمودارهای ۳ بعدی در مطلب ابتدا با استفاده از دستور $meshgrid$ بازه مربوط به داده‌های x و y مشخص می‌شود. سپس محور z بر حسب x و y تعریف می‌شود. در نهایت با استفاده از دستور $mesh(x, y, z)$ نمودار به صورت ۳ بعدی رسم می‌شود.

در هنگام کد نویسی با متلب ممکن است با مشکلات و سوالات مختلفی روبرو شوید. *help* متلب ابزار مفیدی برای یافتن پاسخ به سوالات و حل کردن مشکلات مختلف است. با باز کردن محیط متلب در گوشه سمت راست و بالای صفحه محیط مستطیل شکل برای جستجو را مشاهده می کنید. با جستجو کردن کلیدواژه خود در این قسمت می توانید درباره کلید واژه جستجو شده اطلاعات بیشتری استخراج کنید. برای کد نویسی در متلب به دستورات مختلف مانند *if* و *for* و *while* احتیاج خواهد داشت. با نحوه کدنویسی به وسیله این دستورات به کمک ابزار *help* در متلب می توانید آشنا شوید.

۲.۲.۱ آشنایی با کد نویسی در پایتون

پایتون یکی از زبان های برنامه نویسی پرکاربرد در حوزه هوش است. سرعت اجرای بالا یکی از مزیت های این زبان برنامه نویسی است.

در این قسمت مروری کلی بر کد نویسی در محیط پایتون انجام می شود. در ابتدا با چگونگی تعریف تابع در پایتون آشنا خواهیم شد سپس مروری کوتاه بر روی آرایه ها در پایتون انجام می شود با گذر از این موضوع به ترسیم نمودار در پایتون پرداخته می شود. این بخش با معرفی کردن ابزار *help* در پایتون پایان می پذیرد.

تعریف تابع

تابع در پایتون به وسیله کلید واژه *def* تعریف می شود. بعد از نوشتن کلید واژه اسم تابع و سپس ورودی های تابع مشخص می شود. بعد از مشخص کردن ورودی با قرار دادن : می توانید بدنه تابع خود را بنویسید. در نهایت پارامتر خروجی تابع به وسیله *return* مشخص می شود.

شکل ۳.۱ چگونگی تعریف تابع در پایتون را نمایش می دهد.

```
def functionname( parameters ):
    "function_docstring"
    function_suite
    return [expression]
```

شکل ۳.۱: تعریف تابع در پایتون

آرایه ها و ماتریس ها

یکی از کتابخانه های مفید در پایتون ، کتابخانه *numpy* است. آرایه ها و ماتریس ها با استفاده از این کتابخانه در محیط پایتون تعریف می شوند.

برای استفاده از کتابخانه ابتدا با استفاده از دستور *numpyimport* کتابخانه مورد نظر را وارد کد خود کنید. سپس با استفاده

از دستور `np.array()` می توانید آرایه ها و ماتریس های خود را ایجاد کنید.
شکل ۱.۴ یک مثال از چگونگی تعریف آرایه ها در پایتون را نشان می دهد. همان طور که در شکل نمایش داده شده است اعضای آرایه به وسیله ”،“ از یکدیگر جدا می شوند.

شکل ۱.۵ یک مثال از تعریف ماتریس در پایتون را نشان می دهد. همان طور که در شکل نمایش داده شده است در درون کروشه اصلی هر یک از سطرهای ماتریس در یک کروشه مشخص می شود. اعضای هر سطر با استفاده از ”،“ از یکدیگر جدا می شوند.

```
np.array([1, 2, 3])
```

شکل ۴.۱: تعریف آرایه در پایتون

```
np.array([[1, 2], [3, 4]])
```

شکل ۵.۱: تعریف ماتریس در پایتون

ترسیم نمودار

در پایتون نیز همانند مطلب امکان ترسیم نمودارهای ۲ بعدی و ۳ بعدی وجود دارد.
برای رسم نمودار در پایتون می توانید از کتابخانه `matplotlib` استفاده کنید. برای استفاده از این کتابخانه لازم است در ابتدا کتابخانه `import` را `import matplotlib` کنید. برای `import` کردن کتابخانه کافی است دستور `import matplotlib` را اجرا کنید.
بعد از `import` کردن کتابخانه با استفاده از دستور `matplotlib.plot(x, y)` می توانید داده های خود را رسم کنید. همانند مطلب x آرایه ای از داده متناظر بر روی محور افقی و y آرایه ای از داده های متناظر با هر x می باشد.

راهنمای پایتون

برای استفاده از ابزار راهنمای پایتون برای آشنایی بیشتر با این زبان برنامه نویسی می توانید (`help()`) را در کنسول واقع شده در سمت راست محیط برنامه نویسی وارد کنید. این تابع کلیدواژه مورد جستجو را به عنوان ورودی دریافت می کند.

۳.۱ شرح آزمایش

در این قسمت به پیاده سازی یکتابع ساده در محیط مطلب و پایتون پرداخته می شود.
توجه کنید تابع یک بار در محیط پایتون و یک بار در محیط مطلب باید پیاده سازی شود.

تابع *sigmoid* و مشتق آن از پرکاربرد ترین توابع در شبکه عصبی هستند. تابع *sigmoid* و مشتق آن توسط معادلات ۱.۱ توصیف شده اند.

$$f(x) = \frac{2}{1 + \exp(-x)} - 1 \quad (1.1)$$

$$\dot{f}(x) = \frac{2\exp(-x)}{(1 + \exp(-x))^2} = \frac{1 - (f(x))^2}{2}$$

تابعی به اسم *sigmatrix* بنویسید که یک ماتریس و ابعاد آن را از ورودی دریافت کند و دارای ۲ خروجی به صورت زیر باشد. (A_{ij} اعضای ماتریس ورودی را نشان می‌دهد و *sigmoid* مشتق تابع *sigmoid* است. n و m نیز به ترتیب برابر با تعداد سطرها و ستون‌های ماتریس ورودی است):

۱. خروجی اول :

$\sum_{i=1}^n \sum_{j=1}^m \text{sigmoid}(A_{ij})$

برای پیاده‌سازی تابع خواسته شده مراحل زیر را دنبال کنید:

۱. یک تابع بنویسید که یک عدد را از ورودی دریافت کند و مقدار تابع *sigmoid* را در آن نقطه باز گرداند.

۲. تابعی بنویسید که یک عدد را از ورودی دریافت کند و مقدار مشتق تابع *sigmoid* در آن نقطه را باز گرداند.

۳. یک تابع به اسم *sigmatrix* بنویسید که ماتریس و ابعاد آن را از ورودی دریافت می‌کند و برای تمام اعضای ماتریس توابع نوشته شده در قسمت ۱ و ۲ را صدا می‌زند (توسط یک حلقه *for*) سپس توسط عمل جمع خروجی‌های ۱ و ۲ خواسته شده را تولید کنید.

بعد از پیاده‌سازی تابع ماتریس M را که توسط معادلات ۱.۲ توصیف شده است به همراه تعداد سطرها و ستون‌های آن به تابع نوشته شده بدهید و خروجی را مشاهده کنید.

$$M = \begin{bmatrix} 1 & 0 & \sin(\pi/4) \\ 0 & 1 & \sin(\pi/2) \\ 1 & 0 & 1 \end{bmatrix} \quad (2.1)$$

۴.۱ تمرین

۱. تابعی بنویسید که یک ماتریس و ابعاد آن را از ورودی دریافت کند و $\sum_{i=1}^n \sum_{j=1}^m \tanh(A_{ij})$ را باز گرداند. (A نمایش دهنده ماتریس ورودی تابع و n تعداد سطرها و ستون‌های آن هستند) تابع $\tanh(x)$ به صورت زیر تعریف می‌شود.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.1)$$

آزمایش دوم(پیاده سازی پرسپترون)

۱.۲ پیش گزارش

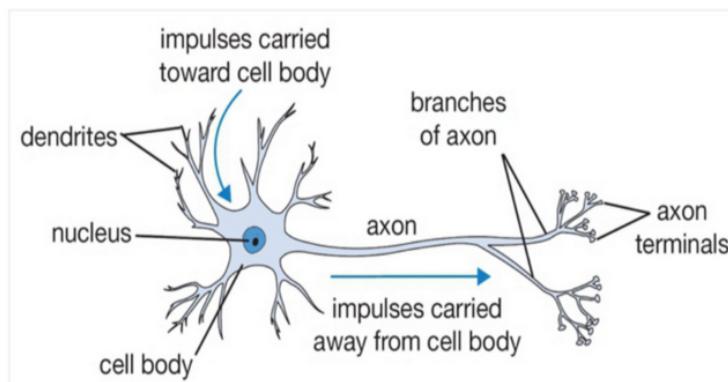
۱. نقش در شبکه عصبی چیست؟
۲. یک مدل ساده از *perceptron* رسم کنید.
۳. شبهایت مدل رسم شده در قسمت ۲ را با ساختار *perceptron* در بدن انسان بیان کنید.

۲.۲ مقدمه

در این آزمایش قصد داریم به پیاده سازی *perceptron* در محیط پایتون بپردازیم.

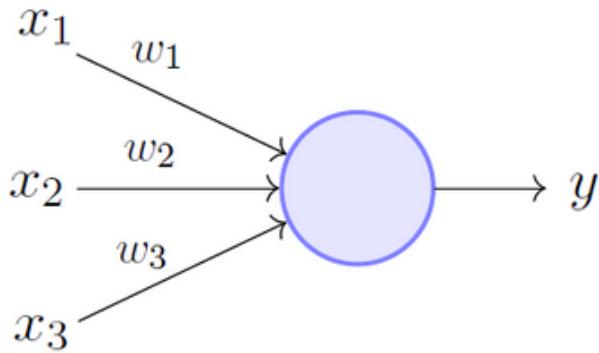
ساختار شبکه عصبی با ساختار مغز انسان شبهایت زیادی دارد.

مغز انسان در خود تعداد بسیار زیادی از نورون‌ها را جای داده است تا اطلاعات مختلف را پردازش کرده و جهان اطراف را بشناسد. به صورت ساده، نورون‌ها در مغز انسان اطلاعات را از نورون‌های دیگر به وسیلهٔ دندرویدها می‌گیرند. این نورون‌ها اطلاعات ورودی را با هم جمع کرده و اگر از یک حد آستانه‌ای فراتر رود به اصلاح فعال می‌شود و این سیگنال فعال شده از طریق آکسون‌ها به نورون‌های دیگر متصل می‌شود. شکل ۱.۲ مدل کلی پرسپترون در مغز انسان را نمایش می‌دهد. با الهام



شکل ۱.۲: مدل پرسپترون در مغز

گیری از مدل پرسپترون در مغز انسان ، مدل پرسپترون در شبکه عصبی ایجاد شد. مدل کلی پرسپترون در شبکه عصبی در شکل ۲.۲ نمایش داده شده است. ورودی پرسپترون مجموع وزن داری از ورودی های اصلی شبکه است و خروجی آن یک مقدار دودویی که نشان دهنده کلاس داده ورودی است ، می باشد.



شکل ۲.۲: مدل پرسپترون در شبکه عصبی

خروجی پرسپترون بر اساس ورودی های آن در معادلات ۱.۲ توصیف شده است.

$$y = \begin{cases} 1 & \text{if } wx > 0 \\ 0 & \text{if } wx < 0 \end{cases} \quad (1.2)$$

وزن های پرسپترون توسط الگوریتم یادگیری پرسپترون تازه سازی می شوند. در ادامه به بررسی این الگوریتم پرداخته می شود.

۱.۲.۲ الگوریتم یادگیری

هدف الگوریتم یادگیری پرسپترون پیدا کردن بهترین وزن های شبکه برای تشخیص درست کلاس داده های ورودی است. الگوریتم یادگیری پرسپترون در شکل ۳.۲ نمایش داده شده است. در این الگوریتم وزن های شبکه در ابتدا با استفاده از اعداد تصادفی مقدار دهی اولیه می شوند. سپس تا هنگامی وزن ها همگرا نشده اند ، وزن ها بر اساس تفات بین کلاس پیش بینی شده توسط پرسپترون و کلاس اصلی داده ورودی تازه سازی می شوند.

۳.۲ شرح آزمایش

در این قسمت به پیاده سازی پرسپترون در محیط پایتون پرداخته می شود.

- Given P training pairs $\{x_1, d_1, x_2, d_2, \dots, x_p, d_p\}$ where x_i is $(n \times 1)$, d_i is (1×1) , $i = 1, \dots, P$
 - The augmented input vectors are $y_i = [x_i \ -1]^T$, for $i = 1, \dots, P$
 - In the following, k is training step and p is step counter within training cycle.
 - Choose $\eta > 0$, $\lambda = 1$, $E_{max} > 0$
 - Initialized weights at small random values, w is $(n \times 1) \times 1$
 - Initialize counters and error: $k \leftarrow 1$, $p \leftarrow 1$, $E \leftarrow 0$
 - Training cycle begins here. Set $y \leftarrow y_p$, $d \leftarrow d_p$, $o = f(w^T y)$ ($f(\text{net})$ is sigmoid function)
 - Update weights $w \leftarrow w + \frac{1}{2}\eta(d - o)(1 - o^2)y$
 - Find error: $E \leftarrow \frac{1}{2}(d - o)^2 + E$
 - If $p < P$ then $p \leftarrow p + 1$, $k \leftarrow k + 1$, go to step 4, otherwise, go to step 8.
 - If $E < E_{max}$ the training is terminated, otherwise $E \leftarrow 0$, $p \leftarrow 1$ go to step 4 for new training cycle.

شکل ۳.۲: الگوریتم یادگیری پرسپترون

۱.۳.۲ پیاده سازی پرسپترون

برای پیاده سازی پرسپترون یک کلاس کلی به پرسپترون ایجاد کنید و هر یک از اعمال زیر را به صورت یک تابع در کلاس بنویسد:

۱. تابعی بنویسید که وزن و $threshold$ را به صورت رندم مقدار دهی اولیه کنید.

۲. تابعی بنویسید که ورودی x را که معرف هر داده است بگیرد و خروجی شبکه را محاسبه کند

۳. تابعی بنویسید برای یادگیری شبکه مطابق شکل ۲.۲ بنویسید ورودی این تابع x (داده ورودی) و y (خروجی واقعی برای ورودی شبکه) و نرخ یادگیری است

۲.۳.۲ داده ورودی

برای چک کردن عملکرد الگوریتم از سرطان موجود در *sklearn* استفاده کنید.
این دیتاست شامل ۵۶۹ داده و ۳۰ متغیر است. خروجی هر داده به صورت ۰ و ۱ است که نشان دهنده وجود و یا عدم وجود سرطان در شخص مورد نظر است. قسمتی از داده را برای یادگیری و قسمت دیگر را برای تست الگوریتم در نظر بگیرید.

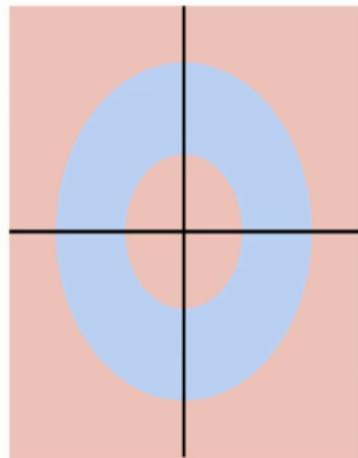
برای عملکرد بهتر الگوی تم بهتر است ابتدا داده ورودی $scale$ کنید و سپس، به صورت ورودی، به شکه داده شود.

۳.۳.۲ بهیود مدل

با تغییر دادن نرخ یاد گیری و نسبت داده یادگیری و تست سعی کنید عملکرد مدل را بهبود ببخشید(دقت به عنوان معیار عملکرد الگو، بتم در نظر گرفته مم شود)

۴.۲ تمرین

۱. در الگوریتم پیاده سازی شده هیچ مقدار بایاسی برای پرسپترون در نظر گرفته نشده است . الگوریتم پیاده سازی شده را با درنظر گرفتن بایاس دوباره پیاده سازی کنید.(برای در نظر گرفتن بایاس کافی است ۱ بعد به ماتریس ورودی خود اضافه کنید و مقدار آن را برابر ۱ قرار دهید). اضافه کردن بایاس چه تاثیری بر روی عملکرد الگوریتم می گذارد؟
۲. فرض کنید میخواهیم کلاس داده هایی به شکل ۴.۲ را توسط پرسپترون پیش بینی کنیم(نقاط صورتی رنگ متعلق به کلاس صفر و نقاط دیگر متعلق به کلاس ۱ در نظر گرفته می شود) عملکرد الگوریتم را پیش بینی کنید؟ آیا الگوریتم پادگیری همگرا می شود؟



شکل ۴.۲: داده ها و کلاس متناظر با آن ها

۳. کد پیاده سازی شده در آزمایشگاه را دوباره در محیط متلب پیاده سازی کنید.

آزمایش سوم(پیاده سازی شبکه عصبی چند لایه)

۱.۳ پیش گزارش

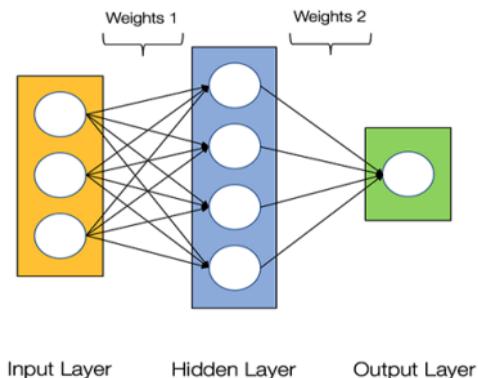
۱. محدودیت های مدل پرسپترون را بیان کنید.
۲. ساختار کلی یک شبکه عصبی ۲ لایه را رسم کنید
۳. مراحل الگوریتم پس انتشار خطا را با جزئیات بیان کنید.

۲.۳ مقدمه

مدل در نظر گرفته شده برای پرسپترون از محدودیت های زیادی رنج می برد برای مثال تنها توابع جداپذیر خطی امکان دسته بندی توسط این نوع مدل را دارند. مدل پرسپترون حتی توانایی پیش بینی تابع ساده XoR را ندارد. شبکه های عصبی چندلایه مدل توسعه یافته پرسپترون هستند و مشکلات مدل پرسپترون را ندارند. در این آزمایش به پیاده سازی یک شبکه عصبی ۲ لایه پرداخته می شود. ساختار کلی یک شبکه عصبی ۲ لایه در شکل ۱.۳ نمایش داده شده است.

عناصر اصلی تشکیل دهنده شبکه عصبی ۲ لایه عبارتند از:

۱. لایه ورودی
۲. لایه مخفی
۳. لایه خروجی
۴. مجموعه ای از وزن ها و بایاس



شکل ۱.۳: ساختار کلی یک شبکه عصبی ۲ لایه

۵. توابع فعال ساز

انتخاب های مختلفی برای توابع فعال ساز یک شبکه عصبی وجود دارد. از توابع فعال ساز پرکاربرد می توان به $tanh$ و $sigmoid$ اشاره کرد

خروجی یک شبکه عصبی ۲ لایه به صورت معادلات ۱.۳ متناظر با تابع فعال ساز شبکه و W_1 و W_2 به ترتیب متناظر با وزن های لایه اول و دوم و b_1 و b_2 به ترتیب متناظر با بایاس لایه اول و دوم در نظر گرفته شده است.

$$\hat{y}(x) = \sigma(W_2\sigma(w_1x + b_1) + b_2) \quad (1.3)$$

برای سنجش عملکرد شبکه یک تابع هزینه در نظر گرفته می شود این تابع معمولاً به صورت توان ۲ خطأ است و به صورت زیر تعریف می شود.

$$cost - function = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.3)$$

هدف شبکه عصبی ۲ لایه پیدا کردن مقدار وزن ها و بایاس است به گونه ای که تابع هزینه در نظر گرفته شده به کمترین مقدار خود برسد. به پیدا کردن مقادیر مناسب وزن ها و بایاس های در شبکه عصبی برای بهبود عملکرد آن، یادگیری شبکه گفته می شود.

هر مرحله یادگیری شامل مراحل زیر است:

۱. محاسبه خروجی شبکه به ازای داده ورودی به این مرحله *forward pass* گفته می شود
 ۲. تازه سازی وزن ها و بایاس های شبکه بر اساس خطأ. به این مرحله پس انتشار خطأ گفته می شود
- در ادامه به مروری کوتاه بر روی الگوریتم پس انتشار خطأ پرداخته می شود.

۱.۲.۳ الگوریتم پس انتشار خطا

مطابق با الگوریتم پس انتشار خطا برای پیدا کردن بهترین وزن ها که منجر به کمینه شدن تابع هزینه می شوند باید در خلاف جهت مشتق تابع هزینه بر حسب وزن ها حرکت کرد.

اگر تابع هزینه را برابر با توان ۲ خطا در نظر گرفته شود (معادله ۲.۳) ، به دلیل اینکه وزن ها در تابع هزینه به صورت مستقیم قرار ندارند ، نمی توان به صورت مستقیم از تابع هزینه بر حسب وزن مشتق گرفت . برای مشتق گرفتن از تابع هزینه بر حسب وزن می توان از قانون زنجیره ای استفاده کرد برای مثال برای تازه سازی وزن های لایه اول شبکه می توان به صورت زیر عمل کرد:

$$\begin{aligned} Loss(y, \hat{y}) &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ \frac{\partial Loss(y, \hat{y})}{\partial w} &= \frac{\partial Loss(y, \hat{y})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} * \frac{\partial z}{\partial w} \quad \text{where } z = Wx + b \\ &= 2(y_i - \hat{y}_i) * \text{derivative of sigmoid function} * x \\ &= 2(y_i - \hat{y}_i) * z(1-z) * x \end{aligned}$$

وزن های لایه دوم نیز به روش مشابه تازه سازی می شوند.

۳.۳ شرح آزمایش

در این قسمت به پیاده سازی یک شبکه عصبی ۲ لایه در محیط پایتون پرداخته می شود.

۱.۳.۳ پیاده سازی شبکه عصبی ۲ لایه

برای پیاده سازی شبکه عصبی ۲ لایه یک کلاس به اسم *deep network* ایجاد کنید و مراحل زیر را دنبال کنید.

۱. یک تابع در کلاس بنویسید که مقدار دهی اولیه وزن ها و بایاس های لایه اول و دوم را انجام دهد(مقدار دهی اولیه را می توانید در تابع سازنده کلاس نیز انجام دهید)

۲. تابعی بنویسید که خروجی لایه اول شبکه را محاسبه کند(تابع *sigmoid* را به عنوان تابع فعالساز در نظر بگیرید)

۳. یک تابع برای محاسبه *sigmoid* پیاده سازی کنید

۴. یک تابع برای محاسبه مشتق تابع *sigmoid* پیاده سازی کنید.

۵. یک تابع برای محاسبه ضرب داخلی وزن ها و ورودی شبکه بنویسید

۶. با استفاده از توابعی که پیاده سازی کرده اید، تابعی بنویسید که وزن های لایه اول و دوم را تازه سازی کند.

۲.۳.۳ تست مدل

همان طور که می دانید تابع xor را نمی توان با یک پرسپترون ساده تخمین زد . در این قسمت برای ارزیابی مدل خود از تابع xor استفاده کنید.

ورودی و خروجی های تابع xor در شکل ۲.۳ نمایش داده شده است . قسمتی از داده را برای یادگیری و قسمتی دیگر را برای تست در نظر بگیرید. ورودی و خروجی ها را با استفاده از آرایه ها به مدل پیاده سازی شده خود دهید.
دقت مدل پیاده سازی شده بر روی داده تست و یادگیری چه قدر است؟

X1	X2	X3	Y
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

شکل ۲.۳: تابع xor

۴.۳ تمرین

۱. مدل شبکه عصبی ۲ لایه پیاده سازی شده در محیط پایتون را دوباره در محیط متلب تکرار کنید.

آزمایش چهارم (پیاده سازی الگوریتم

$(k - mean$

۱.۴ پیش گزارش

۱. مفهوم الگوریتم با نظارت و بدون نظارت را بیان کند.
۲. مزیت های الگوریتم های با نظارت را نسبت به الگوریتم های بدون نظارت و بر عکس بیان کنید.
۳. الگوریتم دسته بندی $k - mean$ را با جزئیات توضیح دهید.
۴. کاربرد های الگوریتم $k - mean$ را بنویسید.

۲.۴ مقدمه

در آزمایشات گذشته با استفاده از شبکه عصبی چند لایه به دسته بندی داده های با برچسب پرداخته شد. در این آزمایش قصد داریم به دسته بندی داده های بدون برچسب بپردازیم. به یادگیری بر روی داده های بدون برچسب و تلاش برای پیدا کردن الگوهای نهفته در آن ها یادگیری بدون نظارت می گویند. یکی از پرکاربردترین الگوریتم ها در یادگیری بدون نظارت الگوریتم $k - mean$ است. در ادامه به بررسی روند اجرای الگوریتم $k - mean$ پرداخته می شود.

۱.۲.۴ الگوریتم $k - mean$

- الگوریتم $k - mean$ یک الگوریتم بازگشتی است که با یک فرض اولیه درباره مراکز دسته ها آغاز می شود. در هر مرحله از اجرای الگوریتم مراحل زیر اجرا می شود:
۱. پیدا کردن دسته متناظر با تمام نقاط

۲. تازه سازی مراکز دسته ها

در مرحله اول اجرای الگوریتم فاصله تمام نقاط تا مراکز دسته ها محاسبه می شود و سپس هر داده متعلق به دسته ای که کمترین فاصله با آن را دارد می شود.

بعد از پیدا کردن دسته های متناظر با هر داده ، در مرحله دوم میانگین داده های متعلق به یک دسته به عنوان مرکز دسته در نظر گرفته می شود.

مراحل ۱ و ۲ تا زمانی که مراکز دسته ها تغییر نکند و یا تغییرات خیلی کمی داشته باشند ادامه می یابد.
بعد از ثابت شدن مراکز دسته ها الگوریتم همگرا می شود.

روند کلی اجرای الگوریتم در شکل ۱.۴ نمایش داده شده است

K-MEANS(P, k)

Input: a dataset of points $P = \{p_1, \dots, p_n\}$, a number of clusters k
Output: centers $\{c_1, \dots, c_k\}$ implicitly dividing P into k clusters

```
1 choose  $k$  initial centers  $C = \{c_1, \dots, c_k\}$ 
2 while stopping criterion has not been met
3   do ▷ assignment step:
4     for  $i = 1, \dots, N$ 
5       do find closest center  $c_k \in C$  to instance  $p_i$ 
6         assign instance  $p_i$  to set  $C_k$ 
7   ▷ update step:
8   for  $i = 1, \dots, k$ 
9     do set  $c_i$  to be the center of mass of all points in  $C_i$ 
```

شکل ۱.۴: مراحل اجرای الگوریتم $k - mean$

در زمان تست دسته ای که به داده ورودی کمترین فاصله را دارد به داده ورودی نسبت داده می شود.

۳.۴ شرح آزمایش

در این قسمت ابتدا به پیاده سازی الگوریتم $k - mean$ پرداخته می شود . بعد از پیاده سازی الگوریتم به تست و بررسی کاربردها و معایب الگوریتم پرداخته می شود.

۱.۳.۴ پیاده سازی الگوریتم

در شکل ۱.۴ مراحل اجرای الگوریتم $k - mean$ نمایش داده شده است . با توجه به مراحل اجرای الگوریتم به پیاده سازی الگوریتم بپردازید. دقت کنید پیاده سازی باید در محیط پایتون انجام شود و کد پیاده سازی شده باید مستقل از تعداد کلاس و نوع داده ورودی باشد ، به بیانی دیگر کد پیاده سازی شده باید به ازای هر داده ورودی و هر تعداد کلاس کارآیی

داشته باشد(تعداد کلاس ها و داده را از ورودی دریافت کنید.)

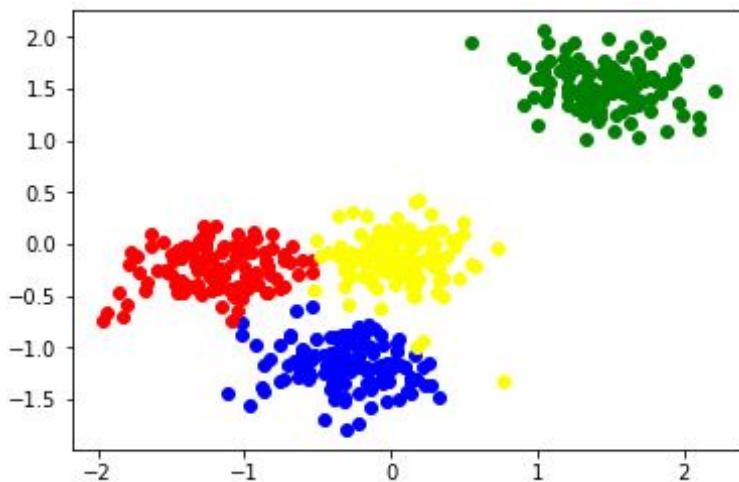
برای پیاده سازی الگوریتم مراحل زیر را انجام دهید:

۱. تابعی بنویسید که مراکز دسته ها را با استفاده از اعداد تصادفی مقدار دهی اولیه کند. ورودی این الگوریتم تعداد کلاس ها و خروجی آن مقدار اولیه مراکز دسته ها است.
۲. تابعی بنویسید که یک داده و مراکز دسته ها را به عنوان ورودی بگیرد و اندیس نزدیکترین دسته به داده ورودی در آرایه را برگرداند.
۳. تابعی بنویسید که آرایه ای از داده ها را از ورودی دریافت کند و سپس میانگین داده های ورودی را باز گرداند.
۴. با استفاده از توابع پیاده سازی شده الگوریتم $k - mean$ را پیاده سازی کنید

۲.۳.۴ تست الگوریتم

مجموعه ای از داده ها در اختیارتان قرار می گیرد.

الگوریتم را بر روی مجموعه داده هایی که در اختیارتان قرار گرفته است با تعداد دسته های برابر با ۲ و ۳ و ۴ اجرا کنید و دسته های خروجی را مشاهده کنید. دسته های خروجی را به رنگ های مختلف مانند (شکل ۲.۴) رنگ آمیزی کنید.



شکل ۲.۴: دسته بندی انجام شده توسط الگوریتم $k - mean$

۳.۳.۴ ارزیابی الگوریتم

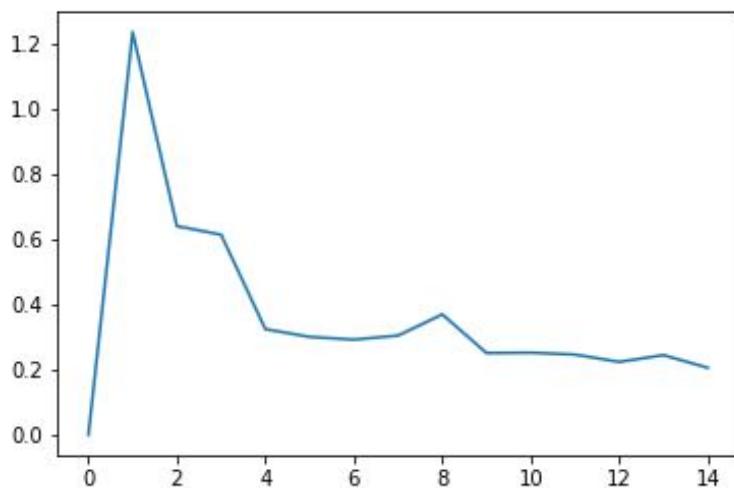
الگوریتم $k - mean$ در دسته الگوریتم های بدون نظارت قرار میگیرد.

تعداد دسته ها و شرایط اولیه برای مراکز دسته ها در عملکرد این الگوریتم تاثیر فراوانی می گذارند برای ارزیابی الگوریتم $k - mean$ مراحل زیر را دنبال کنید:

۱. تابعی بنویسید که نقاط و دسته بندی متناظر با آن ها را به از ورودی بگیرد، سپس برای هر دسته میانگین فاصله نقاط متعلق به دسته مورد نظر را تا مرکز دسته محاسبه کند به این عدد خطای هر دسته گفته می شود.

۲. تابعی بنویسید که نقاط و دسته بندی متناظر با آن ها را به از ورودی بگیرد و میانگین خطای دسته ها را به عنوان خطای الگوریتم محاسبه کند و به عنوان خروجی باز گرداند.

بعد از پیاده سازی تابع بر روی مجموعه داده هایی که در اختیارتان قرار داده شده است، الگوریتم $k - mean$ را به ازای دسته های ۱ تا ۱۵ اجرا کنید و در هر مرحله خطای الگوریتم را محاسبه کنید. در انتها نمودار خطا بر حسب تعداد دسته ها مانند شکل ۳.۴ رسم کنید.



شکل ۳.۴: نمودار خطای خروجی الگوریتم بر حسب تعداد دسته ها و محور عمودی خطا را تشنان می دهد.)

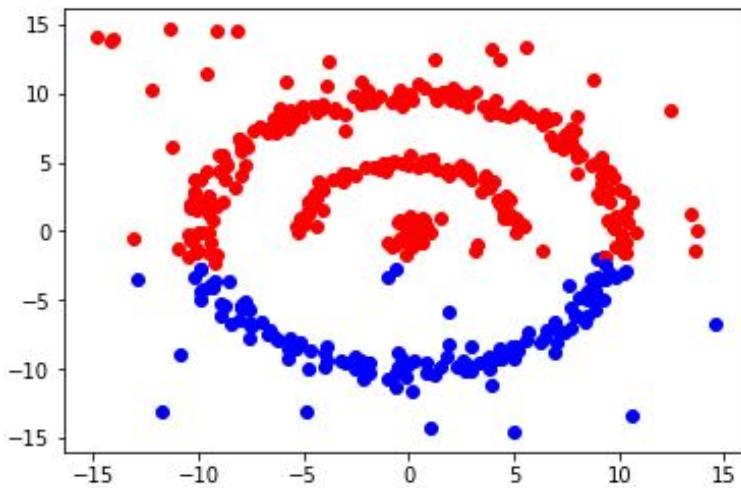
۴.۳.۴ محدودیت های $kmean$

در این قسمت به بررسی محدودیت های الگوریتم $k - mean$ پرداخته می شود. مجموعه ای از داده ها در اختیارتان قرار داده می شود. الگوریتم $k - mean$ پیاده سازی شده را بر روی مجموعه داده ای که در اختیارتان قرار داده شده با تعداد دسته های ۴، ۳، ۲ اجرا کنید و دسته های مختلف را رنگ آمیزی کنید و نمودار خطا بر حسب تعداد دسته ها را رسم کنید.

در صورت پیاده سازی صحیح الگوریتم خروجی مشابه تصویر ۴.۴ خواهد بود.

۴.۴ کاهش حجم عکس به وسیله $k - mean$

یکی از کاربردهای الگوریتم $k - mean$ کم کردن حجم عکس است. یک عکس در اختیارتان قرار داده می شود. در این قسمت قصد داریم با استفاده از الگوریتم $k - mean$ پیاده سازی شده به



شکل ۴.۴: محدودیت های $kmean$ (محورها متناظر با ویژگی های داده های ورودی است)

کم کردن حجم عکس بپردازیم.
برای کاهش حجم عکس مراحل زیر را دنبال کنید:

۱. با استفاده از دستور $imread$ عکس *filename* را از پوشه مجموعه داده (این پوشه در اختیارتان قرار داده می شود) وارد محیط پایتون کنید (با بارگذاری عکس متغیر مربوط به آن در محیط پایتون یک آرایه ۳ بعدی با تعداد سطرها و ستون های ۸۰۰ خواهد بود. سطرها و ستون ها متناظر با مکان پیکسل ها خواهد بود و هر بعد آرایه متناظر با مقادیر R و G و B است).
۲. برای استفاده از الگوریتم $k - mean$ که پیاده سازی کرده اید ابتدا آرایه ۳ بعدی عکس را به یک آرایه ۲ بعدی تبدیل کنید در آرایه دو بعدی هر سطر متناظر با یک پیکسل خواهد بود.
۳. الگوریتم $k - mean$ را با تعداد کلاس برابر با ۱۶ اجرا کنید سپس مقدار هر پیکسل را با مقدار مرکز دسته متناظر جایگذاری کنید. عکس خروجی را رسم کنید و با عکس اولیه مقایسه کنید

۵.۴ تمرین

۱. با توجه به نمودار خطاب بر حسب تعداد کلاس ها که در قسمت ارزیابی الگوریتم رسم کرده اید حالت بهینه الگوریتم در چه تعداد کلاس رخ می دهد؟
۲. به نظر شما چرا الگوریتم توانایی دسته بندهی صحیح داده ها را در قسمت محدودیت های $k - mean$ نداشت؟ برای بهبود الگوریتم چه پیشنهادی دارید؟
۳. الگوریتم $k - mean$ پیاده سازی شده را در محیط متلب مجددا پیاده سازی کنید.
۴. دسته بندهی داده های قسمت اول آزمایش را با استفاده از دستور $kmean$ در متلب (دستور آماده متلب) تکرار کنید.

آزمایش پنجم(پیاده سازی شبکه عصبی

(RBF

۱.۵ پیش گزارش

۱. ساختار یک شبکه عصبی RBF را رسم کنید.

۲. شبکه عصبی چند لایه چه تفاوت هایی با شبکه عصبی RBF چیست؟

۲.۵ مقدمه

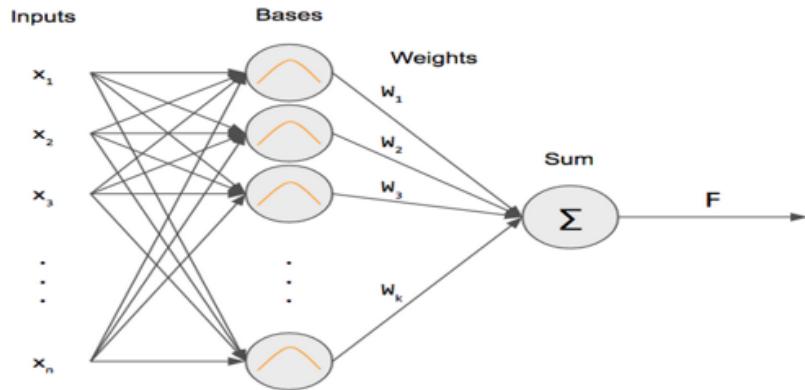
یکی از کاربردهای شبکه عصبی در مساله رگرسیون است. رگرسیون کاربرد های زیادی در پیش بینی بورس و پیش بینی توابع ناشناخته شده دارد. در این آزمایش با استفاده از شبکه عصبی RBF به تخمین یکتابع پرداخته می شود. شبکه های عصبی RBF از شناخته شده ترین و پرکاربرد ترین شبکه های عصبی برای مساله رگرسیون هستند. اگر تعدادی داده از رفتار یکتابع ناشناخته وجود داشته باشد شبکه عصبی RBF با استفاده از ترکیب چندتابع گوسین رفتار تابع ناشناخته شده را با خطای کم پیش بینی خواهد کرد.

۱.۶.۵ ساختار شبکه

ساختار یک شبکه RBF تا حد زیادی به شبکه عصبی چندلایه^۱ شباهت دارد. همان طور که در شکل ۱.۵ مشاهده می کنید. شبکه RBF نیز همانند شبکه تمام پیوسته از ۳ لایه تشکیل شده است. شبکه عصبی RBF دارای ساختار زیر است:

۱. لایه اول ، لایه ورودی است

^۱fully – connected



شکل ۱.۵: ساختار کلی شبکه *RBF*

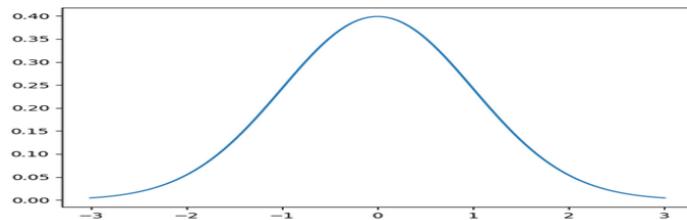
۲. لایه دوم شامل یک یا چند لایه مخفی است

۳. در نهایت لایه آخر که لایه خروجی است.

تفاوت شبکه عصبی *RBF* با شبکه عصبی چند لایه در توابع استفاده شده در لایه مخفی است. در شبکه عصبی *RBF* از توابع گوسین به عنوان توابع فعالساز برای لایه های مخفی استفاده می شود. در ادامه مروری کوتاه بر توابع گوسین خواهیم داشت.

۲.۲.۵ تابع گوسین

تابع گوسین و یا توابع نرمال یکی از مهم ترین توابع در آمار محسوب می شوند. شکل کلی یک تابع گوسین در تصویر ۲.۵ نمایش داده شده است



شکل ۲.۵: تابع نرمال

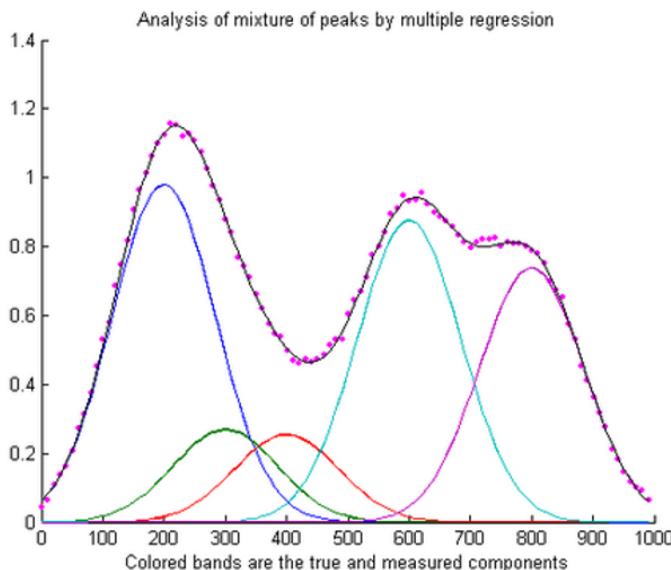
یک تابع گوسین توسط مقدار میانگین و انحراف معیار توصیف می شود. رابطه توصیف کننده تابع گوسین به صورت زیر است:

$$N(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (1.5)$$

در ادامه به بررسی کاربرد توابع گوسین در ساختار شبکه عصبی *RBF* پرداخته می شود.

۳.۲.۵ توابع گوسین و RBF

در شبکه عصبی RBF خروجی به صورت مجموعی از تعدادی تابع گوسین است. در شکل شماره ۳.۵ تعدادی تابع گوسین را مشاهده می کنید. توابع گوسین مختلف به وسیله رنگ های مختلف مشخص شده اند. تفاوت این توابع گوسین در وزن و انحراف معیار و مقدار میانگین آن ها است. اگر مجموع توابع گوسین نمایش داده شده را در شکل شماره ۳.۵ در نظر بگیرید یک تابع پیوسته به دست می آید. اگر مقدار میانگین و انحراف معیار توابع گوسین در نظر گرفته شده تغییر کند، مقدار تابع پیوسته ناشی از مجموع آن ها نیز تغییر خواهد کرد. با توجه به این موضوع در یک شبکه عصبی RBF با استفاده از تعدادی داده از رفتار یک تابع، مقادیر میانگین و انحراف معیار مناسب برای هریک از توابع گوسین به دست خواهد آمد. مقادیر انحراف معیار و میانگین به گونه ای به دست خواهند آمد که مجموع توابع گوسین توصیف مناسبی از تابع بر حسب داده های در اختیار ارایه دهد. بعد از به دست آوردن مقادیر میانگین و



شکل ۳.۵: تخمین تابع پیوسته مجموع توابع گوسین

انحراف معیار برای هریک از توابع گوسین می توان مقدار تابع را در هر نقطه دلخواه پیش بینی کرد. برای به دست آوردن مقدار میانگین برای توابع گوسین در نظر گرفته شده روش های مختلفی وجود دارد یکی از این روش ها استفاده از الگوریتم k میانگین بر روی داده های ورودی برای مقداردهی میانگین های توابع گوسین است.

بعد از مشخص کردن میانگین توابع گوسین استفاده شده برای کامل شدن تابع باید مقدار انحراف معیار هر یک از توابع نیز مشخص شود. برای مشخص کردن انحراف معیار نیز روش های مختلفی وجود دارد در یکی از این روش ها انحراف معیار برابر با رابطه زیر در نظر گرفته می شود در این رابطه d بیشترین فاصله بین دو دسته و m مشخص کننده تعداد دسته ها است.

$$\sigma = \frac{d}{\sqrt{2m}} \quad (3.5)$$

مانند دیگر انواع شبکه های عصبی وزن های خروجی شبکه RBF نیز بر حسب خطا تازه سازی می شوند. در ادامه به بررسی قانون تازه سازی وزن ها در شبکه RBF پرداخته می شود.

۴.۲.۵ قانون تازه سازی وزن های شبکه

خروجی کلی یک شبکه عصبی RBF به صورت زیر است:

$$F(x) = \sum_{j=1}^k w_j \phi_j(x, c_j) + b \quad (3.5)$$

در رابطه بالا ϕ نشان دهنده توابع گوسین است و w وزن متناظر با هریک از توابع گوسین را نمایش می دهد. پارامتر b نیز به عنوان بایاس شبکه در نظر گرفته می شود.

همان طور که از رابطه بالا مشخص است مانند بقیه شبکه های عصبی خروجی شبکه عصبی RBF نیز خروجی به صورت مجموع وزن دار تعدادی توابع گوسین خواهد بود. در قسمت قبل روش هایی برای پیدا کردن مقادیر میانگین و انحراف معیار هریک از توابع گوسین بیان شد. برای کامل کردن شبکه باید مقادیر وزن های شبکه نیز تعیین شوند. مانند بقیه شبکه های عصبی برای تازه سازی وزن های هر یک از توابع گوسین از پس انتشار خطا استفاده می شود. برای آشنایی بیشتر با الگوریتم پس انتشار خطا می توانید به *lecture* های درس مراجعه کنید.

۳.۵ شرح آزمایش

در این مرحله به پیاده سازی یک شبکه عصبی RBF در محیط پایتون پرداخته می شود. در انتهای از استفاده از شبکه عصبی طراحی شده قصد داریم به پیش بینی یک تابع بپردازیم.

۱.۳.۵ پیاده سازی شبکه

برای پیاده سازی شبکه RBF مراحل زیر را دنبال کنید:

۱. برای اولین قدم ابتدا یک تابع برای تولید تابع نمایی بنویسید این تابع یک نقطه و انحراف معیار و میانگین تابع نمایی را از ورودی می گیرد و مقدار تابع نمایی در نقطه ورودی را باز می گرداند.
۲. در قدم بعدی تابع k میانگین را برای مقدار دهی اولیه مقادیر میانگین تابع گوسین بنویسید . ورودی این تابع داده های ورودی و تعداد دسته ها است. برای این مرحله می توانید از تابع k میانگین نوشته شده در جلسه گذشته استفاده کنید. در این تابع باید مقادیر میانگین و انحراف معیار هر یک از توابع گوسین نیز مشخص شود.

۳. یک کلاس *RBF* ایجاد کنید این کلاس تعداد توابع *RBF* و نرخ یادگیری را از ورودی دریافت می کند. در تابع سازنده این کلاس وزن ها و بایاس را با استفاده از اعداد تصادفی مقدار دهی اولیه کنید. در این کلاس تابع زیر را پیاده سازی کنید.

(آ) در این کلاس یک تابع به اسم *fit* تعریف کنید. وظیفه این تابع مشخص کردن مقادیر میانگین و انحراف معیار برای توابع گوسین است. با توجه به وظیفه این تابع، برای پیاده سازی این تابع کافی است که تابع *k* میانگین را که در مرحله پیش پیاده سازی کرده اید صدا بزنید.

با توجه به اینکه مقادیر میانگین و انحراف معیار را در تابع *fit* مشخص کرده اید. برای کامل کردن شبکه باید به پیاده سازی پس انتشار خطا برای تازه سازی وزن های خروجی بپردازید.

(ب) یک تابع به اسم *train* در کلاس *RBF* تعیین کنید. وظیفه این تابع پیاده سازی قانون پس انتشار خطا است. برای پیاده سازی الگوریتم پس انتشار خطا می توانید از توابع نوشته شده در جلسات گذشته نیز استفاده کنید. قسمت اصلی کد این تابع در تصویر شماره ۴.۵ نمایش داده شده است.

```

for i in range(X.shape[0]):
    # forward pass
    a = np.array([self.rbf(X[i], c, s) for c, s, in zip(self.centers, self.std
F = a.T.dot(self.w) + self.b

loss = (y[i] - F).flatten() ** 2
print('Loss: {:.2f}'.format(loss[0]))

# backward pass
error = -(y[i] - F).flatten()

# online update
self.w = self.w - self.lr * a * error
self.b = self.b - self.lr * error

```

شکل ۴.۵: الگوریتم پس انتشار خطا

(ج) در نهایت یک تابع برای پیش بینی کردن خروجی تابع تخمین زده شده به ورودی دلخواه بنویسید. برای نوشتن این تابع کافی است رابطه زیر را در تابع پیاده سازی کنید. دقت کنید مقدار *k* به عنوان ورودی در کلاس مشخص شده است. توابع گوسین را نیز در تابع *fit* محاسبه کرده اید.

$$F(x) = \sum_{j=1}^k w_j \phi_j(x, c_j) + b \quad (4.5)$$

۲.۳.۵ تست شبکه

برای بررسی عملکرد شبکه طراحی شده تعداد ۱۰۰ نمونه از یک تابع سینوس را که به نویز سفید آغشته شده است را به عنوان داده ورودی به شبکه بدهید. چگونگی تولید ۱۰۰ نمونه داده سینوسی و افزودن نویز به این داده ها در شکل شماره نمایش داده شده است.

بعد از دادن ورودی به شبکه و صدا زدن تابع متناظر با یادگیری وزن های شبکه، خروجی شبکه را برای ۱۰۰ نمونه بعدی تابع *sin* به دست آورید. نمودار خروجی اصلی و خروجی پیش بینی شبکه را رسم کنید.

```
NUM_SAMPLES = 100
X = np.random.uniform(0., 1., NUM_SAMPLES)
X = np.sort(X, axis=0)
noise = np.random.uniform(-0.1, 0.1, NUM_SAMPLES)
y = np.sin(2 * np.pi * X) + noise
```

شکل ۵.۵: تولید داده تست

۴.۵ تمرین

۱. شبکه عصبی پیاده سازی شده را در محیط متلب مجددا پیاده سازی کنید.

آزمایش ششم (پیاده سازی شبکه عصبی

(Hopfield

۱.۶ پیش گزارش

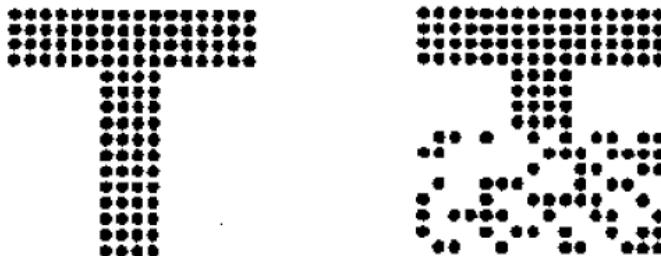
۱. با مراجعه به اینترنت در مورد حافظه‌ی انجمانی در انسان‌ها تحقیق کنید و چکیده‌ی آن را به کلاس ارائه دهید.
۲. با مراجعه به اینترنت در مورد کاربردهای حافظه‌ی انجمانی در علوم مختلف تحقیق کنید و چکیده‌ی آن را به کلاس ارائه دهید
۳. تفاوت حافظه‌ی *Autoassociative* و *Hetroassociative* را با مثالی از حافظه انسان شرح دهید.
۴. درباره‌ی دستور *newhop* در مطلب و کاربرد آن تحقیق کنید. ورودی‌ها و خروجی‌های این تابع را مشخص کنید و با مثالی نحوه‌ی استفاده از این دستور را شرح دهید.
۵. در شبکه‌ی هاپفیلد به روز رسانی سنکرون و ناسنکرون الگوها چه پیامدهایی می‌تواند داشته باشد. توضیح دهید.

۲.۶ مقدمه

فرایند یادگیری و یا به خاطر سپردن اطلاعات همراه با عوامل و احساسات دیگر صورت می‌گیرد. بطور مثال وقتی ما نام یک شهر را به خاطر می‌سپاریم، اسم شهر با خاطرات مختلفی که از آن شهر وجود دارد ذخیره و یا فراخوانی می‌شود. قطعاً همه ما اسم دوستانمان را با تغییر جزئی در ظاهر وی از خاطر نمی‌بریم.

از مهم‌ترین عملکردهای مغز ما، تعیین و فراخوانی حافظه است. حافظه ما در یک حالت انجمانی یا به صورت محتوای آدرس‌پذیر عمل می‌کند. به بیان دیگر حافظه جداگانه موجود نیست و در مجموعه خاصی از نورون‌ها قرار داده شده است. تمامی حافظه‌ها از بعضی جهات، رشتۀ‌ای از حافظه‌ها هستند. برای مثال انسان می‌تواند اشخاص را به طرق مختلف، از جمله رنگ مو یا چشم‌ها، شکل بینی، قد و صدا به خاطر بیاورد و این امکان وجود دارد که با یادآوری یک یا دو ویژگی

فرد، به تمامی حافظه دست پیدا کنیم. شبکه عصبی انجمنی قادر است تا حافظه یا الگوهای خاص را در حالتی مشابه مغز ذخیره نماید. شکل زیر را در نظر بگیرید که در آن کاراکتر T و همان کاراکتر با مقداری نویز مشاهده می‌شود. در صورتی که یک شبکه‌ی انجمنی، کاراکتر T را به خاطر سپرده باشد، با دیدن کاراکتر T نویزی و ناقص، با استفاده از

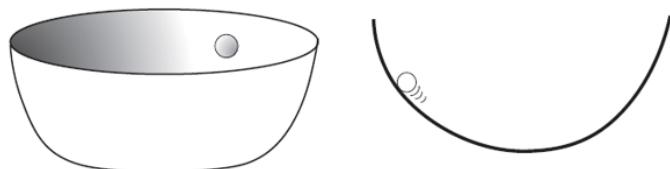


شکل ۱.۶: چپ: کاراکتر T ، راست: کاراکتر T مخدوش شده

سرنخی که قسمت کامل آن به شبکه می‌دهد، قادر است تمام الگوی کاراکتر T را فراخوانی نماید.

۱.۲.۶ تحلیل فیزیکی حافظه

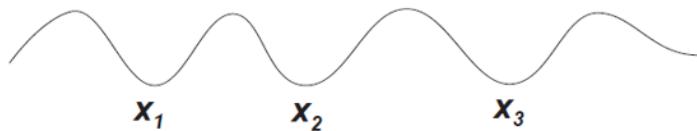
شبکه‌های انجمنی حالت خاصی از یک پدیده‌ی فیزیکی هستند که کاربردهای فراوانی در سایر علوم دارند. شکل زیر یک کاسه به همراه گوی متحرک در آن را نشان می‌دهد:



شکل ۲.۶: گوی و کاسه، تعبیر فیزیکی حافظه‌های انجمنی

هرگاه گوی از هر نقطه‌ی درون کاسه شروع به حرکت کند در نهایت به انتهای کاسه می‌رسد و در آنجا ساکن می‌شود. می‌توان چنین تعبیر نمود که مکان گوی در انتهای کاسه همان داده‌های به خاطر سپرده شده و نقطه‌ی اولیه‌ی گویی همان اطلاعات اولیه برای بازیابی اطلاعات به خاطر سپرده شده است. مشخصه‌ی خاص نقطه‌ی انتهای کاسه، کمینه بودن انرژی پتانسیل آن است.

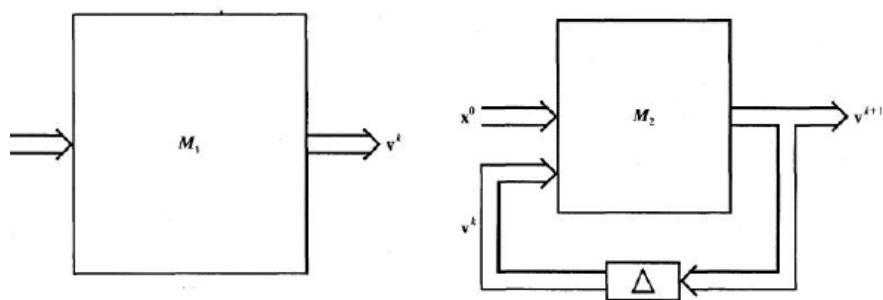
در یک شبکه انجمنی نیز به طریق مشابه یکتابع انرژی برای شبکه تعریف می‌گردد که الگوها در نقاط کمینه‌ی تابع انرژی ذخیره می‌گردند. در شکل ۳.۶، الگوهای x_1, x_2, x_3 ذخیره شده است. به راحتی می‌توان با داشتن نقطه‌ی اولیه، الگوی بازیابی شده را حدس زد.



شکل ۳.۶: ذخیره الگوها در کمینه های محلی

۲.۲.۶ شبکه هاپفیلد

شبکه های انجمانی به دو دسته‌ی کلی استاتیک و دینامیک تقسیم می‌شوند. در حافظه‌ی استاتیک به محض اعمال ورودی، خروجی آماده می‌گردد و هیچ فیدبکی بین ورودی و خروجی وجود ندارد. اما در شبکه های دینامیکی، بعد از اعمال ورودی، شبکه طی یک فرایند دینامیکی و با استفاده از فیدبک خروجی، الگوها را فراخوانی می‌نماید. شکل زیر نحوه عملکرد ۲ شبکه‌ی استاتیکی و دینامیکی را نمایش می‌دهد: رابطه‌ی بین ورودی و خروجی شبکه‌ی استاتیکی بصورت



شکل ۴.۶: راست: حافظه دینامیکی چپ: حافظه استاتیکی

رابطه زیر است:

$$\begin{aligned} v^k &= M_1[x^k] \\ v^{k+1} &= M_2[x^k, v^k] \end{aligned} \quad (1.6)$$

۳.۶ شرح آزمایش

۱. یک شبکه هاپفیلد طراحی کنید تا:

(آ) الگوی تک قطبی $x = [1, 1, 0, 0]$ را ذخیره نماید

(ب) الگوی دو قطبی $x = [1, -1, 1, 1]$ را ذخیره نماید.

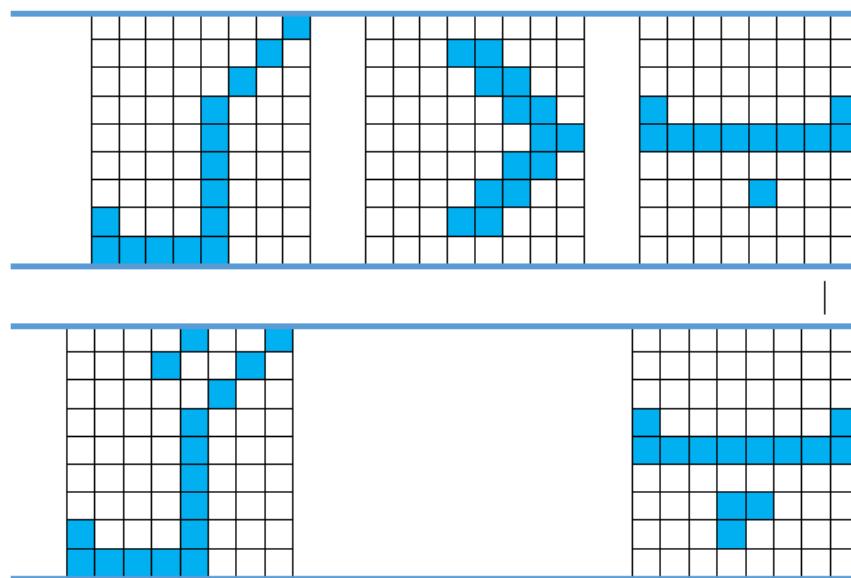
(ج) دو الگوی دو قطبی $x_1 = [-1, 1, 1, 1]$ و $x_2 = [1, 1, -1, 1]$ را ذخیره نماید.

(د) ماتریس W مربوط به ضرایب یک شبکه‌ی عصبی به صورت زیر است که یک الگو را در خود ذخیره کرده است.

برنامه ای بنویسید که الگوی ذخیره شده را بدهد.

$$w = \begin{bmatrix} 0 & 0 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix} \quad (2.6)$$

۲. با استفاده از شبکه‌ی هایپرگراف و دستور *newhop* برنامه‌ای بنویسید که سه حرف "ب", "د" و "ک" را ذخیره کند. سپس حروف ورودی را با ۲۰ درصد نویز جمع کنید و به عنوان ورودی شبکه تزریق کنید. آیا شبکه قادر به بازیابی حروف ذخیره شده است؟



شکل ۵.۶:

۴.۶ تمرین

۱. با استفاده از شبکه‌ی هایپرگراف و دستور *newhop* برنامه‌ای بنویسید که دو حرف "پ", "گ" را ذخیره کند. سپس حروف ورودی را با ۴۰ درصد نویز جمع کنید و به عنوان ورودی شبکه تزریق کنید. آیا شبکه قادر به بازیابی حروف ذخیره شده است؟

آزمایش هفتم (شناسایی به کمک شبکه عصبی)

۱.۷ پیش گزارش

۱. تابع تبدیل سیستم زمان گسسته‌ای به صورت زیر است. تابع تبدیل را در محیط سیمولینک مطلب شبیه سازی کنید

$$H(z) = \frac{-z^2 + 1.9z + .95}{z^3 - .18z^2 + .08z - .08} \quad (1.7)$$

۲. با مراجعه به اینترنت مزایای کنترل کننده‌های هوشمند نظیر کنترل کننده‌ی عصبی بر سایر کنترل کننده‌ها را بیان نمایید.

۳. با مراجعه به اینترنت در مورد کاربردهای شناسایی سیستم دینامیکی تحقیق نمایید.

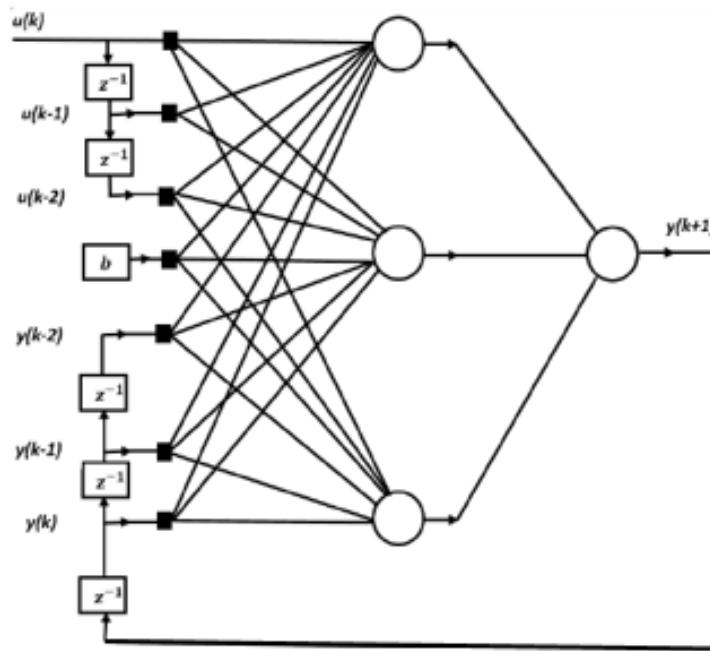
۲.۷ مقدمه

شناسایی سیستم‌ها یکی از پرکاربرد ترین مسایل در علم کنترل است. با استفاده از شبکه‌های عصبی چند لایه‌ی پیاده‌سازی شده در جلسات قبل، می‌توان سیستم‌های ناشناخته را به صورت زمان گسسته یا زمان پیوسته شناسایی کرد. در این آزمایش به شناسایی سیستم‌های گسسته و پیوسته پرداخته می‌شود.

۱.۲.۷ شناساگر استاتیکی

یکی از روش‌های شناسایی برای سیستم‌های دینامیکی استفاده از شناساگر استاتیکی است. در این روش ابتدا تعداد داده ورودی و خروجی از سیستم با تاخیرهای مشخص ذخیره شده و به عنوان ورودی به شبکه عصبی اعمال می‌شود. با اعمال

این داده‌ها، شبکه عصبی آموزش داده شده و مدل را شناسایی می‌نماید. هدف شبکه عصبی تخمین خروجی است به گونه ای که خروجی تخمین زده شده برابر با خروجی اصلی سیستم باشد(یا اختلاف اندکی داشته باشد) در شکل ۱.۷ مدل کلی یک شناساگر استاتیکی نمایش داده شده است. در این شکل ورودی‌های شبکه عصبی ورودی سیستم و ورودی سیستم تاخیر یافته و خروجی سیستم و خروجی سیستم تاخیر یافته است. خروجی شبکه نیز تخمینی از خروجی سیستم است.



شکل ۱.۷: شناساگر استاتیکی

در ادامه به بررسی ساختار شناساگر پیوسته پرداخته می‌شود.

۲.۲.۷ شناساگر پیوسته

مدل‌های شناسایی سیستم پیوسته به ۲ دسته کلی تقسیم می‌شوند:

۱. مدل موازی

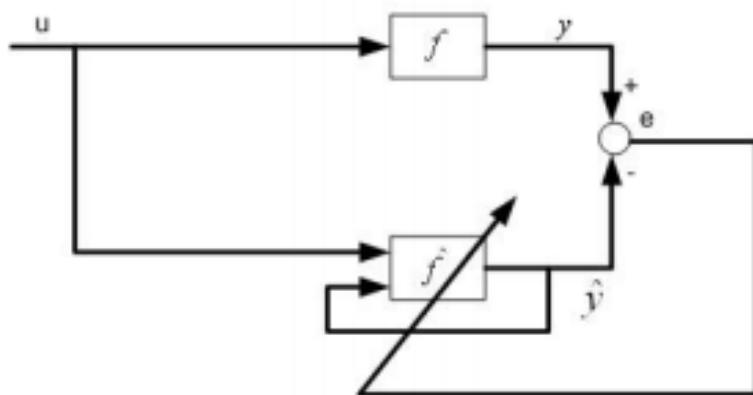
۲. مدل سری-موازی

در ادامه به مروری کلی بر ساختار این ۲ دسته پرداخته می‌شود.

۳.۲.۷ مدل موازی

مدل کلی شناسایی با استفاده از مدل موازی سیستم در شکل ۲.۷ نمایش داده شده است. در این مدل با استفاده از خطای بین خروجی شبکه عصبی و خروجی اصلی سیستم پارامترهای شبکه تازه سازی می‌شوند. مدل‌های مختلفی را می‌توان

برای شبکه عصبی در نظر گرفت برای مثال می توان شبکه عصبی را یک شبکه عصبی چند لایه که وزن های آن با استفاده از قانون پس انتشار خطاطازه سازی می شود در نظر گرفت.



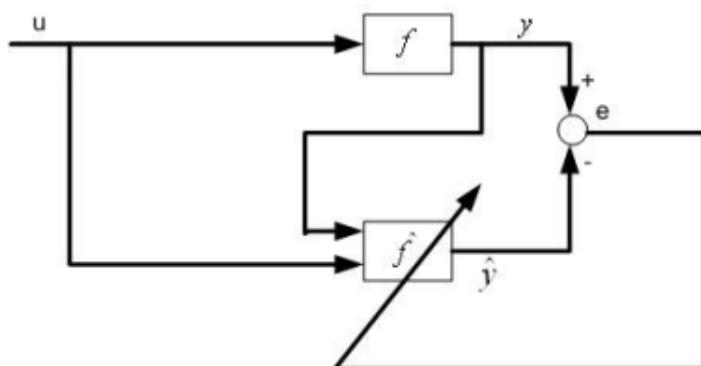
شکل ۲.۷: شناسایی با استفاده از مدل موازی

در مدل موازی فیدبک در نظر گرفته شده در ساختار شبکه عصبی از خروجی خود شبکه است . این فیدبک ممکن است مانع همگرایی شبکه عصبی شود و یا باعث کند شدن سرعت همگرایی شود.

مدل سری-موازی

با فیدبک گرفتن از خروجی اصلی سامانه به جای خروجی شبکه عصبی می توان همگرایی سیستم را نسبت به حالت شناساگر موازی بهبود بخشد.

با تغییر دادن فیدبک در مدل سری (فیدبک موجود در ساختار شبکه عصبی) مدل شناساگر سری-موازی حاصل می شود. ساختار کلی مدل در شکل ۳.۷ نمایش داده شده است. در این مدل نیز پارامترهای شبکه با استفاده از خطای بین خروجی شبکه و خروجی اصلی سیستم تازه سازی می شوند. ورودی این شبکه ورودی اصلی سامانه و خروجی اصلی سیستم است.



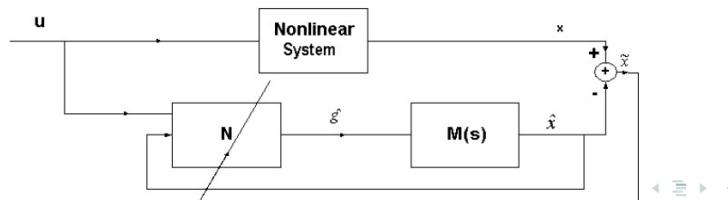
شکل ۳.۷: شناسایی با استفاده از مدل سری-موازی

۴.۲.۷ ترکیب شناساگر استاتیکی و مدل موازی

در مدل استاتیکی ورودی های شبکه شامل خروجی ورودی های تاخیر یافته مدل اصلی سیستم بود. تاخیر در سیستم های پیوسته با استفاده از انتگرال گیر مدل می شود. به دلیل ناپایدار بودن انتگرال گیر از یک فیلتر برای مدل کردن تاخیر در سیستم های پیوسته استفاده می شود. با در نظر گرفتن این نکته برای عملکرد بهتر شناساگر موازی می توان شناساگر موازی را به گونه ای توسعه داد که مقادیر گذشته خروجی نیز در ورودی شبکه وجود داشته باشد.

در این آزمایش به پیاده سازی مدل توسعه یافته شناساگر موازی برای تخمین حالت های یک سیستم پیوسته پرداخته می شود.

ساختار کلی این شبکه در شکل ۴.۷ نمایش داده شده است. در این شکل ورودی شبکه با \hat{w} نمایش داده شده است. ورودی و خروجی سیستم بعد از گذشتن از یک فیلتر است. با توجه به اینکه ورودی شبکه \hat{w} خروجی ها و ورودی های تاخیر یافته نیز در ورودی شبکه تاثیر می گذارند و باعث عملکرد بهتر مدل موازی می شوند. شبکه عصبی را نیز می توان به صورت یک شبکه عصبی چند لایه که با استفاده از قانون پس انتشار خطا وزن های آن تازه سازی می شوند در نظر گرفت. برای آشنایی بیشتر با این مدل و بررسی دقیق تر آن می توانید به lecture های درس مراجعه کنید.¹



شکل ۴.۷: شناساگر موازی توسعه یافته

۳.۷ شرح آزمایش

در این قسمت ابتدا به پیاده سازی یک شناساگر استاتیکی می پردازید سپس به تخمین حالت های یک سیستم پیوسته توسط شناساگر موازی (ترکیب شناساگر استاتیکی و شناساگر موازی) می پردازید

۱.۳.۷ پیاده سازی شناساگر استاتیکی

در این قسمت به پیاده سازی یک مدل ساده از شناساگر استاتیکی پرداخته می شود. برای پیاده سازی یک مدل ساده از شناساگر استاتیکی مراحل زیر را دنبال کنید:

۱. مدل زیر را در محیط سیمولینک متلب شبیه سازی کنید

$$H(z) = \frac{-z^2 + 1.9z + .95}{z^3 - 1.8z^2 + .08z - .08} \quad (2.7)$$

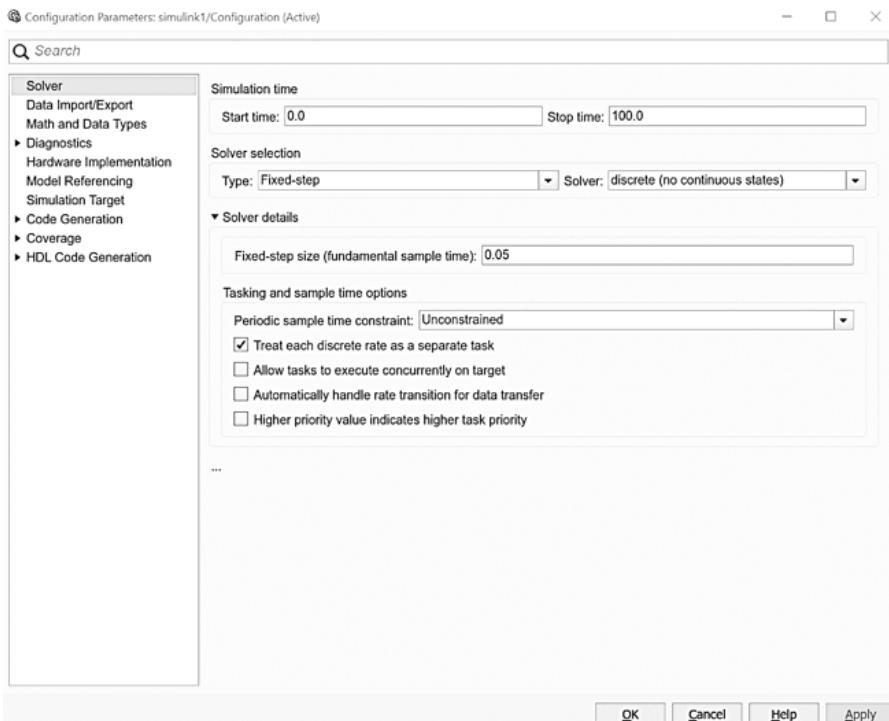
¹امکان ترکیب مدل سری-موازی و مدل استاتیکی نیز وجود دارد که در این گزارش به آن پرداخته نشده است.

۲. نویز سفید را به عنوان ورودی به سیستم شبیه سازی شده اعمال کنید.

۳. ورودی و خروجی سیستم شبیه سازی شده را در *work space* ذخیره کنید.

برای ذخیره کردن ورودی و خروجی ها مراحل زیر را دنبال کنید:

(آ) برای ذخیره ورودی و خروجی های مدل طراحی شده، بایستی اول تنظیمات سیمولینک به گونه ای تغییر داده شود تا امکان ذخیره سازی فراهم شود. برای این منظور از منوی *Model Configuration Parameters*, گزینه *Simulation*, *Solver* را انتخاب کنید. حال در منوی *Solver*, تنظیمات را مطابق شکل ۷.۵ انجام دهید.



شکل ۷.۵: تنظیمات سیمولینک

(ب) با قرار دادن بلوک *simout* در ورودی و خروجی سامانه و اجرای سیمولینک داده های ورودی و خروجی را ذخیره کنید

۴. در پنل *Workspace* مقادیر ورودی و خروجی های تولید شده قابل مشاهده اند. یک *mfile* مطلب باز کنید تا مقادیر مربوطه را در متغیرهای *input* و *output* ذخیره کنید.

۵. در مرحله‌ی بعد لازم است تا داده‌های با تاخیر ذخیره شوند. به این ترتیب می‌توان با محاسبه‌ی داده‌های تاخیردار در ورودی و خروجی، آموزش را بهبود بخشد. (تعداد تاخیر ها در ورودی و خروجی برای هر سیستم با توجه به درجه سیستم متفاوت است).

داده های با تاخیر را می توان به صورت زیر تولید کرد.

$$inputDelayed1 = [0; input(1 : end - 1)];$$

$$inputDelayed2 = [0; 0; input(1 : end - 2)];$$

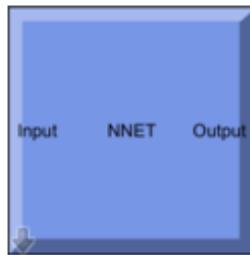
$outputDelayed1 = [0; output(1 : end - 1)];$

$outputDelayed2 = [0; 0; output(1 : end - 2)];$

۶. ورودی شبکه عصبی را به صورت زیر تعریف کنید.

$x=[input, inputDelayed1, inputDelayed2, outputDelayed2, outputDelayed1];$

۷. حال با استفاده از ابزار *FunctionFittingNeuralNetwork* داده‌های جمع آوری شده را مورد آموزش قرار دهید. آموزش را با داده‌های موجود انجام داده و در نهایت شبکه عصبی طراحی شده را به محیط سیمولینک *export* کنید. شبکه عصبی به شکل زیر خواهد بود. که لازم است ورودی‌هایی مشابه آنچه در $inputDelayed_i$ ها ($i = 1, 2$) تعریف شد، در سیمولینک تعریف شده و به سیستم اعمال شود



شکل ۶.۷: بلوک شبکه عصبی استاتیک طراحی شده

تست شناساگر استاتیکی

برای تست مدل طراحی شده، یک ورودی سینوسی به مدل اصلی و مدل تهیه شده به وسیله شبکه عصبی) دهید و خروجی‌ها را مقایسه کنید. (دقت کنید ورودی‌های و خروجی‌های تاخیر دار نیز باید به عنوان ورودی به بلوک شبکه عصبی استاتیک طراحی شده داده شوند. برای دادن چند ورودی به شبکه می‌توانید از *demux* استفاده کنید.)

۲.۳.۷ پیاده سازی شناساگر پیوسته

در این قسمت قصد داریم به تخمین حالت‌های یک سیستم توسط شناساگر موازی توسعه یافته (ترکیب شناساگر موازی و شناساگر استاتیکی) بپردازیم.

در (شکل ۷.۷) مدل کلی شناساگر آورده شده است. برای پیاده سازی این لایه‌ها از شبکه عصبی *MLP* که پیش‌تر طراحی شده بود استفاده می‌شود. طرح کلی از ۳ بخش اصلی تشکیل شده است:

۱. بخش *sigmoid*

۲. بخش خطی

۳. بخش فیلتر

در این مدل ابتدا قوانین تازه سازی وزن های شبکه مطابق با معادله ۷.۳ پیاده سازی می شود . وزن های شبکه بر اساس (خطای شبکه) تازه سازی می شوند. سپس با استفاده از وزن های شبکه \hat{w} (۲ بلوک اول شکل ۷.۷ این قسمت را پیاده سازی می کنند) مطابق با رابطه ۷.۴ محاسبه می شود و درنهایت با استفاده از \hat{w} حالت های سیستم تخمین زده می شوند. (در بلوک سوم شکل ۷.۷ این قسمت پیاده سازی می شود) حالت های تخمین زده شده توسط شبکه عصبی مطابق با رابطه ۷.۵ تازه سازی می شود

$$\dot{\hat{W}} = -\eta_1(\tilde{x}^T A^{-1})^T (\sigma(\hat{V}\hat{x}))^T - P_1 \parallel \tilde{x} \parallel \hat{w}$$

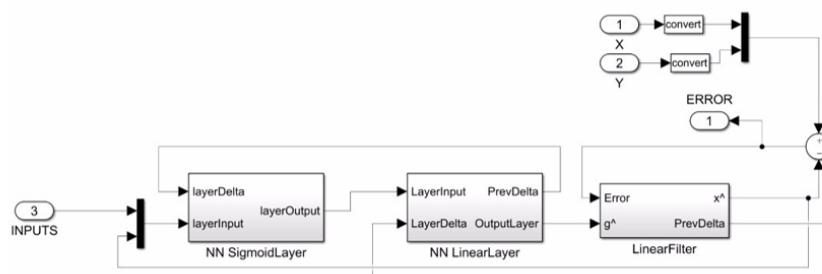
$$\dot{\hat{V}} = -\eta_2(\tilde{x}^T A^{-1} \hat{W}(I - \dot{\sigma}(\hat{V}\hat{x})))^T \hat{x}^T - p_2 \parallel \tilde{x} \parallel \hat{V} \quad (\text{3.7})$$

$$\tilde{x} = x - \hat{x}$$

$$\hat{g}(\hat{x}, u) = \hat{W}\sigma(\hat{V}\hat{\hat{x}}) \quad (\text{F.V})$$

$$\dot{\hat{x}} = A\hat{x} + \hat{g}(\hat{x}, u) + \epsilon \quad (\textcircled{A.V})$$

برای پیاده سازی شبکه مراحل زیر را دنبال کنید. (دقت کنید در مدل پیاده سازی شده ابعاد ورودی و تعداد نورون ها در ادامه به بررسی و پیاده سازی هر یک از بلاک ها پرداخته می شود.



شكل ٧.٧: ساختار کلی شناساگر عصبی

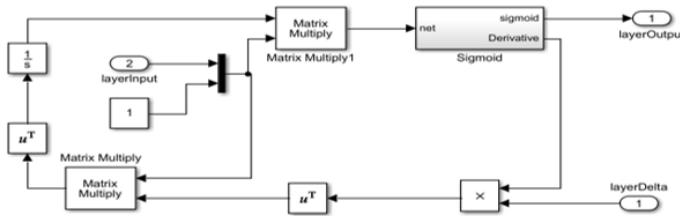
پاید به صورت متغیر تعریف شوند تا در صورت نیاز تغییر پیدا کند):

۱. ابتدا بلوک مربوط به *sigmoid*, *A* بیاده سازی کنید.

ورودی بخش $sigmoid$ ورودی مطلوب سیستم و خطای لایه بعد است مدل کلی بخش $sigmoid$ در شکل ۷.۸ آورده شده است. در این قسمت تابع $sigmoid$ و مشتق آن نیز که برای قوانین تازه سازی لازم است پیاده سازی می شوند.

در این بلاک قوانین تازه سازی \hat{W} مطابق با معادله ۳.۷ پیاده سازی کنید برای پیاده سازی فرض کنید \tilde{x}, W را از ورودی دریافت می کنید. W در بلوک دوم تولید می شود و سپس به صورت فیدبک به این بلوک منتقل می شود.

بعد از پیاده سازی قوانین تازه سازی \hat{V} با استفاده از انتگرال گیری سیگنال \hat{V} را تولید کنید. سپس با پیاده سازی تابع $sigmoid$ مقدار $\sigma(\hat{V}\hat{x})$ را به عنوان خروجی به لایه بعد انتقال دهید. از خروجی این لایه در لایه بعد برای تازه سازی \hat{W} استفاده می شود.

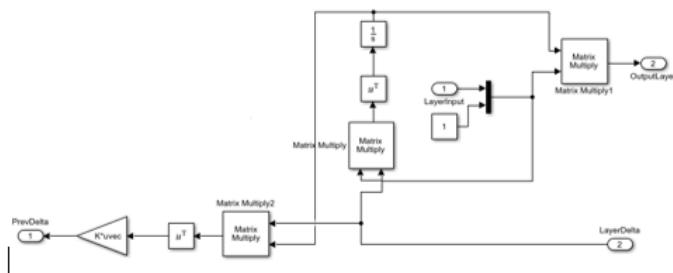


شکل ۸.۷: ساختار کلی بخش $sigmoid$

۲. ساختار کلی این لایه در شکل ۹.۷ نمایش داده شده است.

در این لایه ابتدا با استفاده از خروجی لایه قبل $(\sigma(\hat{V}\hat{x}))$ قوانین تازه سازی \hat{W} را با استفاده از رابطه ۳.۷ پیاده سازی کنید در این قسمت نیز مانند بخش $sigmoid$ فرض کنید. \hat{x} از ورودی دریافت می شود. سپس با انتگرال گیری از قانون تازه سازی \hat{W} ، \hat{W} را تولید کنید. بعد از محاسبه \hat{W} و با استفاده از خروجی لایه قبل $(\sigma(\hat{V}\hat{x}))$ \hat{g} را مطابق رابطه ۴.۷ محاسبه کنید.

با توجه به اینکه W در رابطه تازه سازی \hat{V} مورد نیاز است و با توجه به اینکه قانون تازه سازی \hat{V} در بلوک اول (بخش $sigmoid$) پیاده سازی شده است. \hat{W} را به صورت فیدبک از این بلوک به بلوک اول منتقل کنید. در نهایت با استفاده از \hat{W} و $\sigma(\hat{V}\hat{x})$ (ورودی شبکه) و با توجه به رابطه ۴.۷ \hat{g} را تولید کنید و به عنوان ورودی به لایه بعد انتقال دهید.



شکل ۹.۷: ساختار کلی بخش خطی شناساگر

۳. در این بخش با استفاده از \hat{g} تولید شده در لایه پیشین و با توجه به رابطه ۷.۵ ابتدا \hat{x} را تولید کنید و سپس با انتگرال گیری از \hat{x} سیگنال \hat{x} را تولید کنید.

سپس با کم کردن حالت های تخمین زده شده شبکه از حالت های اصلی سیستم \hat{x} را تولید کنید. با توجه به اینکه این سیگنال را در بلوک های دوم و سوم به عنوان ورودی فرض کردید این سیگنال را به عنوان ورودی به بلوک های دوم و سوم دهید.

۳.۳.۷ تست شناساگر شبیه سازی شده

برای تست شناساگر شبیه سازی شده تابع تبدیل نشان داده شده در معادله ۶.۷ را در محیط سیمولینک متلب پیاده سازی کنید و حالت های سیستم را با استفاده از شناساگر پیاده سازی شده تخمین بزنید. ورودی سیستم را $\sin(t)$ در نظر بگیرید نمودار خطای بین حالت های تخمین زده شده و حالت های اصلی سیستم را رسم کنید.

$$H(s) = \frac{40}{0.02s^2 + s} \quad (6.7)$$

۴.۷ تمرین

- فرض کنید در بازوی ربات با مدل زیر (M) شناخته شده نیست، این سیستم را شناسایی کنید. می خواهیم انتهای بازوی ربات مسیر $\sin 10t$ را دنبال کند.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= M^{-1}(x)[u - C(x_1, x_2)x_2 - Dx_2 - g(x_1)] \end{aligned} \quad (7.7)$$

که $x_1 = [q_1, q_2]^T$ موقعیت انتهای هر لینک $x_2 = \dot{x}_1$ سرعت انتهای هر لینک u گشتاور، M ماتریس اینرسی، C نیروهای کریولیس و گریز از مرکز، D دمپینگ ویسکوز و g نیروی گرانش است و به صورت زیر تعریف شده است:

$$\begin{aligned} M &= \begin{bmatrix} a_1 + 2a_4 \cos q_2 & a_2 + a_4 \cos q_2 \\ a_2 + a_4 \cos q_2 & a_3 \end{bmatrix}, & D &= 0 \\ C &= a_4 \sin q_2 \begin{bmatrix} -\dot{q}_2 & -(\dot{q}_1 + \dot{q}_2) \\ \dot{q}_1 & 0 \end{bmatrix}, & g &= \begin{bmatrix} b_1 \cos q_1 + b_2 \cos(q_1 + q_2) \\ b_2 \cos(q_1 + q_2) \end{bmatrix} \end{aligned}$$

$$a_1 = 200.01, \quad a_2 = 23.5, \quad a_3 = 122.5, \quad a_4 = 25, \quad b_1 = 784.8, \quad b_2 = 245.25$$

آزمایش هشتم (کنترل به کمک شبکه عصبی)

۱.۸ پیش گزارش

۲. روش کنترل به وسیله دینامیک معکوس را با جزئیات بیان کنید.

۳. معاوی روش کنترل به وسیله دینامیک معکوس را بیان کنید.

راهکارهایی برای مقابله با معاوی روش کنترل به وسیله دینامیک معکوس بیان کنید.

۲.۸ مقدمه

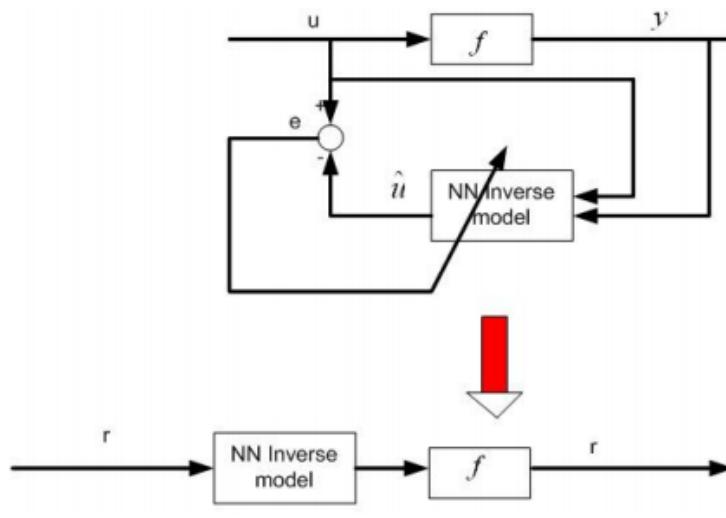
یکی از کاربردهای شبکه عصبی در کنترل یک سیستم ناشناخته است.

در این بخش به کنترل یک سیستم ناشناخته به وسیله شبکه عصبی پرداخته می شود.

یکی از روش های کنترل ، کنترل به وسیله دینامیک معکوس است . در ادامه به مروری کوتاه بر کنترل به وسیله دینامیک معکوس پرداخته می شود.

۱.۲.۸ کنترل دینامیک معکوس

ساختار کلی این روش در شکل نمایش داده شده است. در این روش ابتدا سیستم معکوس سیستم اصلی در صورت ناشناخته بودن شناسایی می شود.(قسمت بالای تصویر). با استفاده از یک شبکه عصبی می توان سیستم معکوس را شناسایی کرد(آزمایش هفتم). سپس با توجه به اینکه $f^{-1}f = I$ با سری کردن مدل معکوس و مدل اصلی خروجی سامانه ، خروجی مرجع را دنبال می کند.(قسمت پایین تصویر)



شکل ۱.۸: کنترل دینامیک معکوس

۳.۸ شرح آزمایش

در این آزمایش به کنترل یک سیستم ناشناخته با استفاده از دینامیک معکوس پرداخته می‌شود.
مراحل زیر را دنبال کنید:

۱. ابتدا سیستم زیر را در محیط سیمولنک متلب شبیه سازی کنید. قصد داریم این سیستم را با استفاده از کنترل دینامیک معکوس، کنترل کنیم. فرض می‌شود سیستم ناشناخته است. (از این مدل تنها برای ایجاد داده ورودی و خروجی برای شبکه عصبی استفاده می‌شود.)

$$H(s) = \frac{40}{s+1} \quad (1.8)$$

۲. با توجه با ساختار شناساگر پیاده سازی شده در آزمایش هفتم و با وارد کردن خروجی سامانه به عنوان ورودی شناساگر (به جای ورودی اصلی سامانه)، سیستم معکوس را شناسایی کنید. (توجه کنید برای ایجاد خطأ در ساختار شناساگر باید از اختلاف بین ورودی سامانه و خروجی شناساگر استفاده کنید)

۳. ساختار پیاده سازی شده در قسمت ۲ (شناسایی سیستم معکوس) را با ساختار سیستم اصلی (تابع تبدیل سیستم) سری کنید.

۴. $\sin(t)$ را به عنوان ورودی به سیستم پیاده سازی شده (در قسمت ۳) اعمال کنید و خطای خروجی را رسم کنید.

۴.۸ تمرین

۱. به کمک مدل پیاده سازی شده در آزمایشگاه سعی کنید سیستم زیر را به وسیله دینامیک معکوس به گونه ای کنترل کنید که خرجی $\sin(2t)$ را دنبال کند.

$$H(s) = \frac{1}{s - 1} \quad (۲.۸)$$

در صورت عملکرد نامناسب مدل ، علت را توضیح دهید.

پروژه بخش عصبی

۱.۹ پیش گزارش

۱. با مراجعه به اینترنت معادلات دینامیک مربوط به خودرو بدون سرنشین چهار چرخ را بیان کنید.
۲. با مراجعه به اینترنت معادلات دینامیک مربوط به ربات نقاش ۲ بعدی را بیان کنید.

۲.۹ مقدمه

در آزمایش های گذشته به بررسی انواع مختلف شبکه عصبی و طراحی کنترلر و شناساگر با استفاده از شبکه عصبی پرداخته شد . در این آزمایش قصد داریم به پیاده سازی سخت افزاری الگوریتم های بیان شده بپردازیم. برای پیاده سازی سخت افزاری از خودرو زمینی چهار چرخ بدون سرنشین^۱ و یا ربات نقاش^۲ بعدی استفاده می شود. در ادامه به معرفی کوتاه ربات نقاش ۲ بعدی و خودرو زمینی بدون سرنشین پرداخته می شود.

۱.۲.۹ خودرو زمینی بدون سرنشین

خودرو زمینی بدون سرنشین یک ربات^۴ چرخ با مکانیزم حرکت تانکی و خودکار بوده که قابلیت طی مسیر به صورت خودکار و ارتباط با عامل های دیگر از طریق بی سیم را نیز دارد. این ربات مجهز به حلقه ۶ تایی سونار برای مانع یابی و همچنین ۲ عدد انکوادر جهت مکان یابی بوده و قابلیت نصب سنسورهای مختلفی در آن در نظر گرفته شده است. توضیحات مفصل این ربات در (پیوست ۱) انجام شده است. (حتما پیش از آزمایش، پیوست مطالعه شود). تصویر این ربات در شکل نمایش داده شده است.

¹ UGV(unmanned – ground – veichle)



شکل ۱.۹: خودروی زمینی بدون سرنشین

۲.۲.۹ ربات نقاش ۲ بعدی

ربات نقاش یک ربات دو بعدی با دو درجه آزادی ، همانطور که در شکل مشاهده می شود، این ربات از چهار لینک (بازو) که از یک طرف به دو محرک (موتور) DC و از طرف دیگر به یکدیگر متصل شده اند تشکیل شده است. نقطه اتصال لینک ها به هم سر موثر نامیده می شود که با اعمال فرمان مناسب به موتورها ، سر موثر به نقاط مطلوب در صفحه $x-y$ حرکت داده می شود. سر موثر شامل یک قلم میباشد که این قلم جهت بر جای گذاشتن مسیر حرکت سر موثر به کار رفته است.



شکل ۲.۹: ربات نقاش ۲ بعدی

۳.۹ شرح آزمایش

در این بخش ۳ آزمایش برای پیاده سازی بر روی ربات نقاش ۲ بعدی و خودرو بدون سرنشین در نظر گرفته شده است . از بین این ۳ آزمایش ۱ آزمایش را انتخاب کرده و به پیاده سازی سخت افزاری آن بپردازید.

۱.۳.۹ آزمایش ۱ - شناسایی خودرو زمینی بدون سرنشین

در این قسمت قصد داریم با استفاده از ساختار شناساگر معرفی شده در آزمایش ۷ به شناسایی خودروی زمینی بدون سرنشین بپردازیم.

برای شناسایی خودرو زمینی بدون سرنشین مراحل زیر را دنبال کنید:

۱. برای پیاده سازی شناساگر لازم است ابتدا از صحت شناساگر طراحی شده در آزمایش ۷ اطمینان حاصل کنید. تابع تبدیل ساده شده خودرو بدون سرنشین به صورت رابطه ۱.۹ است. ابتدا شناساگر طراحی شده در آزمایش ۷ را بر روی این مدل شبیه سازی کنید و از صفر شدن خطای بین خروجی شناساگر و خروجی اصلی سامانه اطمینان حاصل کنید.

$$H(s) = \frac{40}{0.02s^2 + s} \quad (1.9)$$

۲. فایل سیمولینک مربوط به خودرو بدون سرنشین در اختیارتان قرار داده می شود. فایل را در محیط سیمولینک باز کنید.

۳. pwm_1 و pwm_2 را برابر با تابع $\sin(t)$ قرار دهید.

۴. بعد از اطمینان از صحت شناساگر پیاده سازی شده در آزمایش ۷ با استفاده از شناساگر طراحی شده و در نظر گرفتن pwm_1 و pwm_2 خودروی بدون سرنشینی به عنوان ورودی شناساگر و موقعیت خودروی بدون سرنشین به عنوان خروجی (موقعیت در راستای x و یا y) سعی کنید سیستم را شناسایی کنید.

۵. خطای بین خروجی شناساگر(موقعیت تخمین زده شده خودرو بدون سرنشین) و موقعیت خودرو بدون سرنشین را رسم کنید.

۲.۳.۹ آزمایش ۲ - شناسایی ربات نقاش ۲ بعدی

برای شناسایی ربات نقاش ۲ بعدی مراحل زیر را دنبال کنید:

۱. ابتدا فایل ربات را که در اختیارتان قرار گرفته است را اجرا کنید. با اجرا کردن این فایل ربات شروع به نقاشی یک دایره خواهد کرد.

۲. این ربات ۲ سیگنال را از ورودی دریافت می کند(سیگنال های u_1, u_2 در فایل سیمولینک). با در نظر گرفتن این ۲ سیگنال به عنوان ورودی شناساگر و زاویه بازوها به عنوان خروجی شناساگر، سیستم را شناسایی کنید. خطای بین خروجی شناساگر و خروجی اصلی سیستم را رسم کنید.

۳.۳.۹ آزمایش ۳ - کنترل خودروی زمینی بدون سرنشین

در این قسمت به پیاده سازی سخت افزاری الگوریتم کنترل شبیه سازی شده (در آزمایش ۸) بر روی خودرو زمینی بدون سرنشین پرداخته می شود.

قصد داریم خودرو بدون سرنشین را به گونه ای کنترل کنیم که بر روی خط $x = 10 = y$ حرکت کند.
برای کنترل خودروی بدون سرنشین مراحل زیر را دنبال کنید:

۱. ابتدا از صحت کنترلر پیاده سازی شده در آزمایش ۸ اطمینان حاصل کنید.

مدل ساده شده خودرو بدون سرنشین به صورت رابطه ۱.۹ است. ابتدا کنترلر طراحی شده در آزمایش ۸ را برروی این مدل شبیه سازی کنید. ورودی مرجع را برابر با یک عدد ثابت در نظر بگیرید. در صورت عملکرد صحیح کنترلر خطای بین خروجی سامانه و ورودی مرجع به سمت صفر میل می کند.

۲. فایل سیمولینک مربوط به کنترل خودرو بدون سرنشین در اختیارتان قرار داده می شود. فایل را در محیط سیمولینک باز کنید.

۳. ورودی مرجع کنترلر را برابر با ۱۰ در نظر بگیرید. مقدار $pwm_1(pwm_2)$ را برابر با خروجی کنترلر عصبی قرار دهید.
موقعیت خودرو بدون سرنشین را نیز به عنوان ورودی کنترلر در نظر بگیرید.
خطای بین ورودی مرجع و موقعیت خودرو بدون سرنشین رارسم کنید.

۴.۹ تمرین

۱. با توجه به خروجی های حاصل شده در بخش عملی ، ضعف های عملکرد الگوریتم پس انتشار خطا را در حالت گستته بیان کنید.

آزمایش نهم (آشنایی با مفاهیم فازی)

۱.۱۰ پیش گزارش

۱. تفاوت دسته بندی ترد^۱ را با دسته بندی فازی بیان کنید.

۲. مفهوم $S - Norm$ در منطق فازی چیست؟

۳. مفهوم $T - Norm$ در منطق فازی چیست؟

۲.۱۰ مقدمه

برای دسته بندی برخی از خصوصیات و اشیا نمی توان یک مرز مشخص در نظر گرفت، فرض کنید میخواهیم جوان یا پیر بودن انسان ها را مشخص کنیم. تمام انسان ها را نمی توان در ۲ دسته پیر و جوان قرار داد برای مثال یک انسان میانسال را در نظر بگیرید چنانی انسانی ۵۰ درصد جوان و ۵۰ درصد پیر محسوب می شود به همین ترتیب ممکن است یک انسان ۳۰ درصد جوان و ۷۰ درصد پیر محسوب شود. این گونه دسته بندی که در آن مرز مشخصی بین دسته ها در نظر گرفته نمی شود دسته بندی و یا منطق فازی گفته می شود. در این آزمایش قصد داریم به پیاده سازی مفاهیم اولیه فازی در محیط متلب بپردازیم. در ابتدا به مروری کوتاه بر مفاهیم اولیه فازی پرداخته می شود.

۱.۲.۱۰تابع عضویت

تابع عضویت یک تابع پیوسته است. خروجی تابع عضویت در بازه $[0, 1]$ قرار دارد. خروجی تابع عضویت نشان دهنده احتمال متعلق بودن ورودی به کلاس در نظر گرفته شده است.

^۱crisp

۲.۲.۱۰ نقطه عبور

نقطه عبور \mathbb{Z} به نقطه ای گفته می شود که مقدار تابع عضویت در آن نقطه برابر با ۵ است.

۳.۲.۱۰ عملگرها در منطق فازی

برای کار با مجموعه ها در منطق فازی لازم است عملیات های اصلی مانند متمم، اشتراک و اجتماع تعریف شوند. در ادامه به بررسی برخی از عملگرهای اصلی در منطق فازی پرداخته می شود.

مکمل

عملگر مکمل عملگری است که $\mu_A(\text{تابع عضویت } A)$ را به تابع عضویت مکمل آن $(\mu_{\bar{A}})$ ارتباط می دهد. در منطق فازی رابطه های گوناگونی برای مکمل یک مجموعه تعریف شده است، از معروف ترین روابط تعریف شده برای مکمل می توان به رابطه *Sugeno* که در معادلات زیر توصیف شده است اشاره کرد. (در رابطه $1.10 \in (-1, \inf)$ در نظر گرفته شده است)

$$c_{\lambda_a} = \frac{1 - \mu_A(x)}{1 + \lambda} \quad (1.10)$$

۴.۲.۱۰ اجتماع

اجتماع \mathbb{Z} مجموعه در منطق فازی با $S-Norm$ دو مجموعه مشخص می شود. مشابه مکمل گیری برای اجتماع \mathbb{Z} مجموعه در منطق فازی نیز تعاریف متفاوتی وجوددارد. یکی از معروف ترین تعاریف $S-Norm$ در منطق فازی تعریف *Dombi* که مطابق با رابطه $2.10 \in (0, \inf)$ تعریف می شود است. (در رابطه $2.10 \in (0, \inf)$ در نظر گرفته شده است)

$$S_\lambda(\mu_A(x), \mu_B(x)) = \frac{1}{1 + ((\frac{1}{\mu_A(x)} - 1)^{-\lambda} + (\frac{1}{\mu_B(x)} - 1)^{-\lambda})^{-\frac{1}{\lambda}}} \quad (2.10)$$

اشتراک

اشتراک \mathbb{Z} مجموعه در منطق فازی با $T-Norm$ دو مجموعه مشخص می شود. مشابه $S-Norm$ برای $T-Norm$ نیز در منطق فازی تعاریف متفاوتی وجوددارد. یکی از معروف ترین تعاریف $S-Norm$ در منطق فازی تعریف *Dombi* که مطابق با رابطه $3.10 \in (0, \inf)$ تعریف می شود است. (در رابطه $3.10 \in (0, \inf)$ در نظر گرفته شده است)

$$T_\lambda(\mu_A(x), \mu_B(x)) = \frac{1}{1 + ((\frac{1}{\mu_A(x)} - 1)^\lambda + (\frac{1}{\mu_B(x)} - 1)^\lambda)^{\frac{1}{\lambda}}} \quad (3.10)$$

۳.۱۰ شرح آزمایش

در این آزمایش قصد داریم ابتدا ۲ مجموعه فازی را در محیط متلب پیاده سازی کنیم ، بعد از پیاده سازی ۲ مجموعه به پیاده سازی عملگرهای فازی مانند مکمل ، $S - Norm$ و $T - Norm$ پرداخته می شود.
برای ایجاد توابع عضویت فازی مراحل زیر را دنبال کنید:

۱. مجموعه جهانی را برابر با $U = 0 : 0.01 : 1$ در نظر بگیرید.

۲. دستور $gaussmf$ در متلب یک تابع عضویت فازی براساس تابع عضویت گوسی تولید می کند. ورودی های دستور $gaussmf$ داده های ورودی و میانگین و انحراف معیار در نظر گرفته شده برای تابع گوسین است.
با استفاده از دستور $gaussmf$ تابع عضویت A را به صورت زیر تعریف کنید:

$$A = gaussmf(U, [0.350]);$$

۳. با استفاده از دستور $gaussmf$ تابع عضویت B را به صورت زیر تعریف کنید:

$$B = gaussmf(U, [0.350]);$$

در ادامه قصد داریم عملیات های مکمل (مکمل $S - Norm Dombi$ و $T - Norm Dombi$) و $(Sugno)$ را پیاده سازی کنیم.
برای پیاده سازی و تست مکمل $Sugno$ مراحل زیر را دنبال کنید:

۱. با در نظر گرفتن رابطه 1.10 تابعی بنویسید که یک تابع عضویت و یک مقدار در بازه $(-\infty, 1)$ در معادله 1.10 را از ورودی دریافت کند و سپس مقدار مکمل آن را (با توجه به رابطه 1.10) بازگرداند.

۲. تابع عضویت A را که در قسمت قبل پیاده سازی کردید به عنوان ورودی به تابع مکمل $Sugno$ دهید و خروجی را رسم کنید.

۳. تابع عضویت B را که در قسمت قبل پیاده سازی کردید به عنوان ورودی به تابع مکمل $Sugno$ دهید و خروجی را رسم کنید.

برای پیاده سازی و تست $S - Norm Dombi$ مراحل زیر را دنبال کنید.

۱. با در نظر گرفتن رابطه 3.10 تابعی بنویسید که یک دو تابع عضویت (μ_A و μ_B) در رابطه 3.10 و یک مقدار در بازه $(0, \infty)$ در معادله 3.10 را از ورودی دریافت کند و سپس مقدار $S - Norm Dombi$ آن را (با توجه به رابطه 3.10) بازگرداند.

۲. توابع عضویت A و B را که در قسمت های قبل پیاده سازی کردید به عنوان ورودی به تابع $S - Norm Dombi$ دهید و خروجی را رسم کنید.

برای پیاده سازی و تست $T - Norm Dombi$ مراحل زیر را دنبال کنید:

۱. با در نظر گرفتن رابطه 2.10 تابعی بنویسید که یک دو تابع عضویت (μ_A و μ_B) در رابطه 2.10 و یک مقدار در بازه $(0, \infty)$ (مقدار λ در معادله 2.10) را از ورودی دریافت کند و سپس مقدار $T - Norm Dombi$ آن را (با توجه به رابطه 2.10) بازگرداند.

۲. توابع عضویت A و B را که در قسمت های قبل پیاده سازی کردید به عنوان ورودی به تابع $T - Norm Dombi$ دهید و خروجی را رسم کنید.

۴.۱۰ تمرین

۱. همان طور که می دانید انواع مختلفی از $S - Norm$ در منطق فازی وجود دارد. یکی از معروف ترین انواع $S - Norm$ است. در رابطه 4.10 $Einstein - sum$ توصیف شده است. تابع عضویت A و B را در محیط متلب پیاده سازی کنید. تابع $Einstein - sum$ را به صورت زیر تعریف کنید.

$$U = 0 : 0.01 : 1;$$

$$A = gaussmf(U, [0.350]);$$

$$B = gaussmf(U, [0.350]);$$

تابع عضویت A و B را به عنوان ورودی به تابع $Einstein - sum$ دهید و خروجی را رسم کنید.

$$S_{es}(\mu_A, \mu_B) = \frac{\mu_A + \mu_B}{1 + \mu_A \mu_B} \quad (4.10)$$

۲. همان طور که می دانید انواع مختلفی از $T - Norm$ در منطق فازی وجود دارد. یکی از معروف ترین انواع $T - Norm$ است. در رابطه 5.10 $Einstein - product$ توصیف شده است. تابع عضویت A و B را در محیط متلب پیاده سازی کنید. تابع $Einstein - product$ را به صورت زیر تعریف کنید.

$$U = 0 : 0.01 : 1;$$

$$A = gaussmf(U, [0.350]);$$

$$B = gaussmf(U, [0.350]);$$

تابع عضویت A و B را به عنوان ورودی به تابع $Einstein - product$ دهید و خروجی را رسم کنید.

$$S_{es}(\mu_A, \mu_B) = \frac{\mu_A \mu_B}{2 - (\mu_A + \mu_B - \mu_A \mu_B)} \quad (5.10)$$

آزمایش دهم(کنترلر فازی)

۱.۱۱ پیش گزارش

۱. برای ایجاد یک سیستم فازی به مجموعه ای از قوانین اگر - آن گاه احتیاج است. به نظر شما چگونه می توان قوانین فازی مناسب برای کنترل یک سیستم را ایجاد کرد؟
۲. در یک سیستم فازی می توان چند قانون مختلف اگر-آنگاه را با یکدیگر ترکیب کرد. به نظر شما چه منطقی برای ترکیب کردن قوانین مختلف در سیستم فازی باید در نظر گرفته شود؟

۲.۱۱ مقدمه

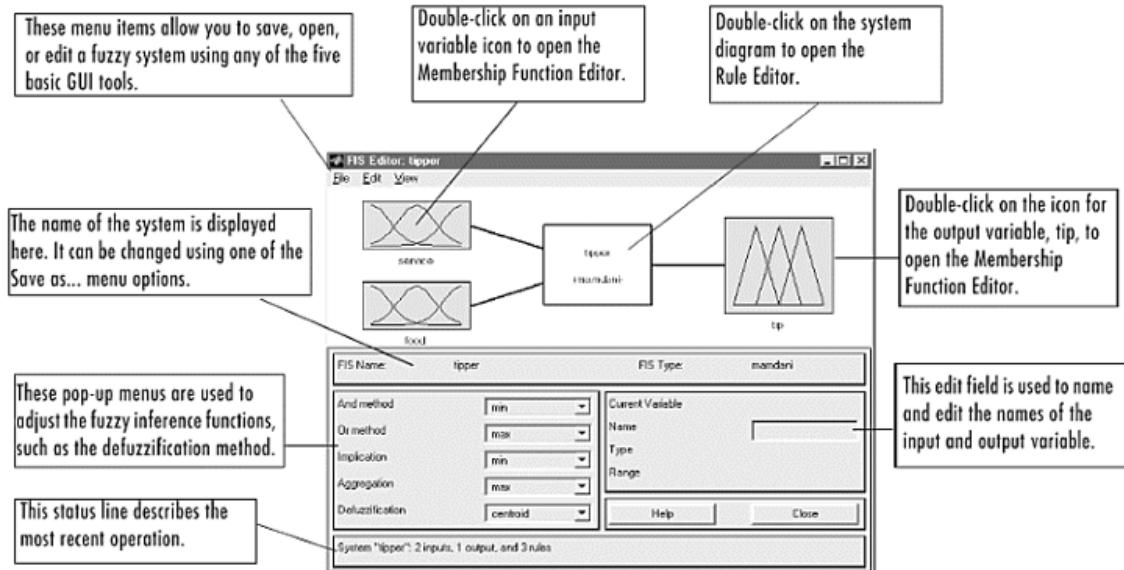
در این آزمایش قصد داریم با استفاده از جعبه ابزار فازی ^۱ متلب به پیاده سازی یک کنترلر فازی بپردازیم. در ابتدا مروری بر چگونگی کارکرد جعبه ابزار فازی متلب خواهیم داشت.

۱.۲.۱۱ جعبه ابزار فازی متلب

در قسمت مربع شکل پایین صفحه همان طور که در شکل ۱.۱۱ می بینید می توان توابع مورد نیاز (*fuzzy – interface*) را مشخص کرد. در قسمت مربع شکل پایین سمتراست می توان نام ورودی و خروجی هارا تغییر داد.

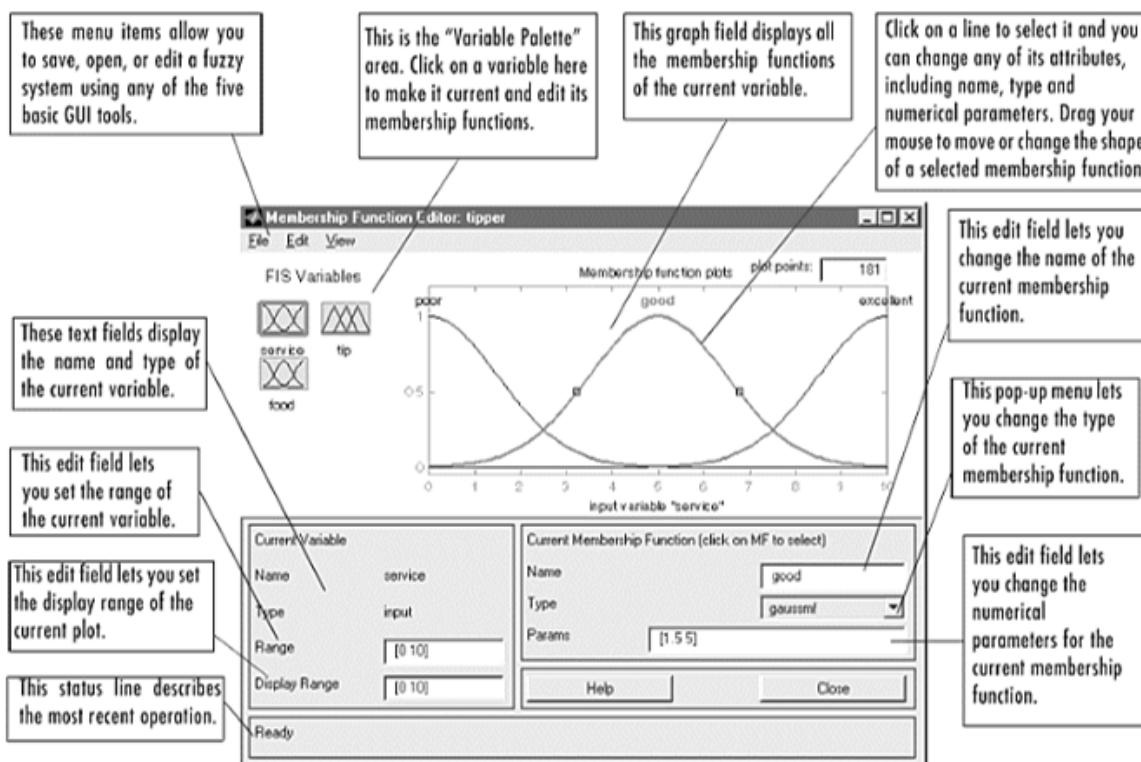
قسمت توابع عضویت پنچره ای همانند شکل ۲.۱۱ دارد. در مربع پایین همانطور که در شکل ۲.۱۱ مشخص است می توان بازه ورودی و خروجی را تعیین کرد. برای تغییر مشخصات هر کدام از خطها ورودی یا خروجی کافی است بر روی آن کلیک کنید.

^۱ If – Then
^۲ fuzzy – toolbox



شکل ۱.۱۱: محیط اصلی ابزار طراحی فازی

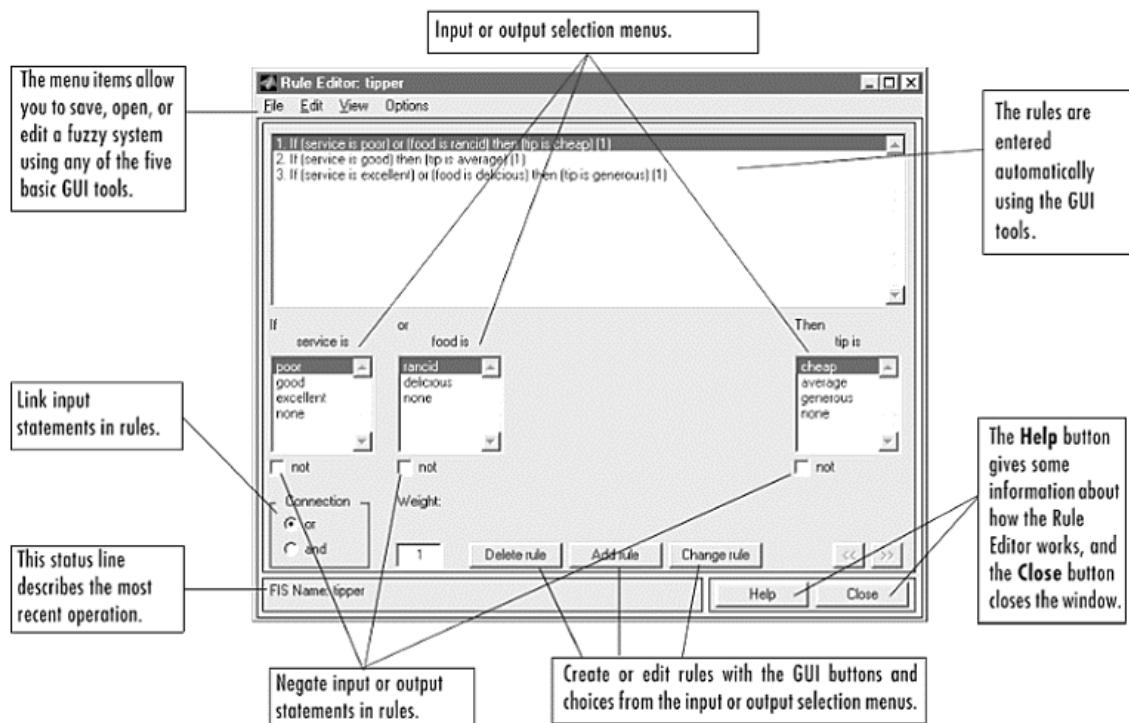
در گراف بالای صفحه نیز می‌توان تمام توابع عضویت را مشاهده کرد. در مربع پایین سمتراست نیز قابلیت تصحیح نام تابع عضویت انتخاب شده را دارد. در این قسمت همچنین می‌توان مقادیر عددی تابع عضویت را تغییر داد. پنچره تنظیم



شکل ۲.۱۱: تعریف تابع عضویت در ابزار فازی متلب

قوانين به صورت شکل ۳.۱۱ است با استفاده از مربعات پایین صفحه به راحتی میتوان قوانین فازی را به صورت اگر-آنگاه پیادهسازی کرد در مربعات سمتراست حالت ورودی با توجه به تابع عضویت تعریف شده و در مربع سمت چپ همان طور

که در شکل ۳.۱۱ نشان داده شده است حالت خروجی با توجه بهتابع عضویت مشخص می‌شود.



شکل ۳.۱۱: تعریف قوانین فازی در ابزار فازی متلب

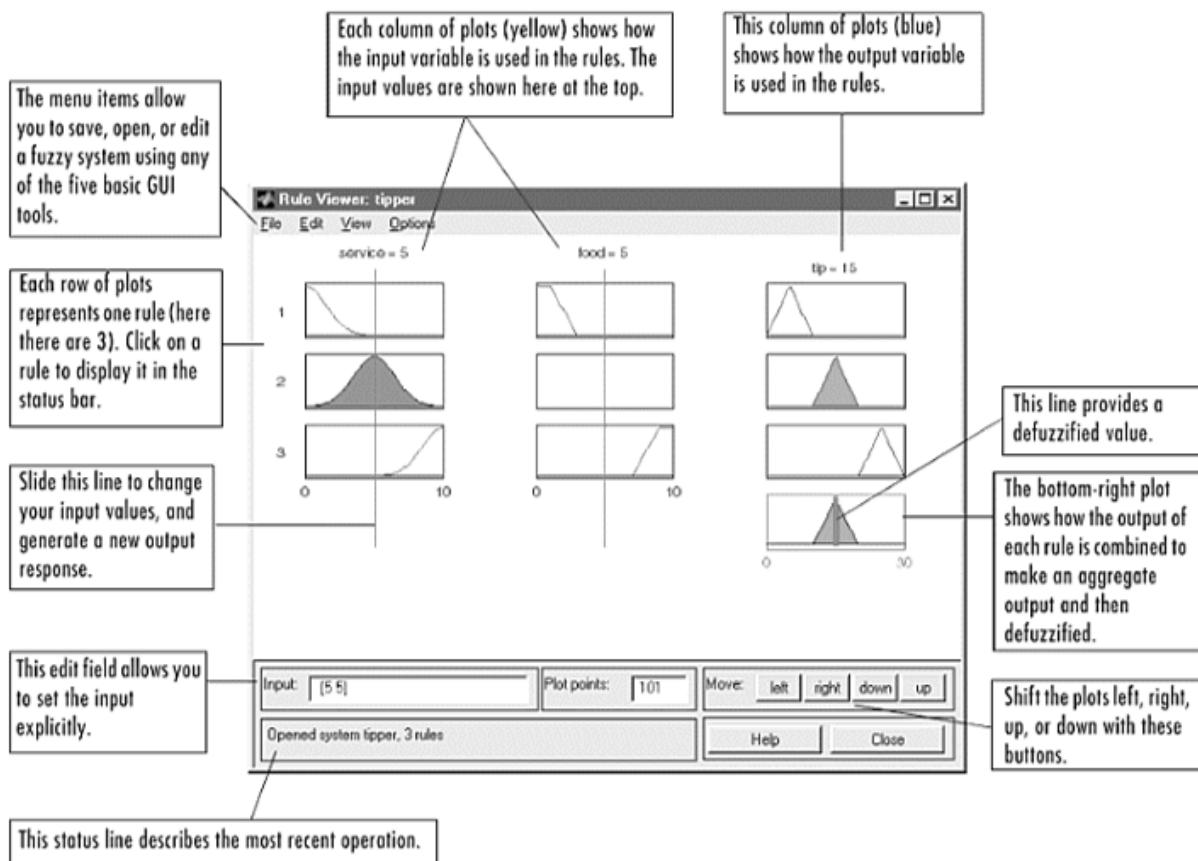
در قسمت *rule – viewer* می‌توان غیرفازی ساز را مانند شکل ۴.۱۱ در ستون سوم مشخص کرد.

۳.۱۱ شرح آزمایش

در این قسمت هدف، طراحی یک کنترلر فازی برای سرعت خودرویی است که دینامیک در حال حرکت آن به صورت زیر تعریف می‌شود.

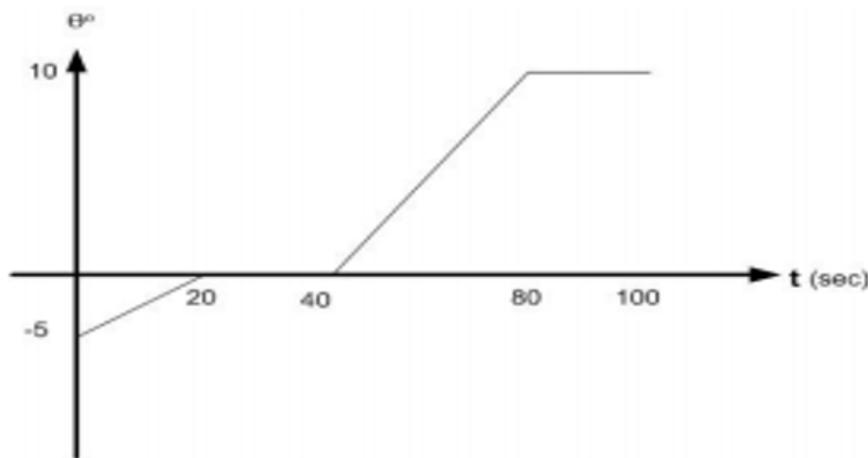
$$\dot{v} = -.1v + [-1, .1][T, \theta]^T \quad (1.11)$$

T و ورودی‌های سیستم بیانگر زاویه پدال گاز و شیب جاده است. هدف طراحی یک کنترلر فازی برای سیستم توصیف شده است به طوری که با گرفتن فیدبک از سرعت کنونی خودرو (v) و با توجه به شیب جاده بتواند سرعت خودرو را در 50 km/h نگه دارد. (ورودی کنترل کننده فازی را خطاب و مشتق خطابی خروجی سیستم در نظر بگیرید). شیب جاده نیز در شکل



شکل ۴.۱۱: مشاهده کردن قوانین در ابزار فازی متلب

شکل ۵.۱۱ توصیف شده است.



شکل ۵.۱۱: نمودار شیب جاده بر حسب زمان

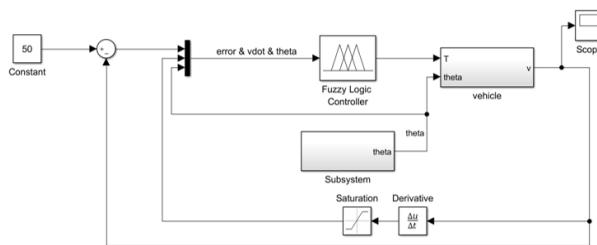
ساختار کلی کنترلر فازی در شکل نمایش ۶.۱۱ داده شده است.

برای طراحی کنترلر فازی مراحل زیر را دنبال کنید:

- تابع عضویت ورودی را تعریف کنید. توجه کنید ورودی دارای ۳ حالت آرام و متوسط و سریع است.

۲. با توجه به ورودی قوانین فازی را تعریف کنید . پیشنهاد می شود برای خروجی فازی ساز ۳ حالت آرام و عادی و سریع در نظر بگیرید.

۳. توابع عضویت خروجی را تعریف کنید



شکل ۱۱: ساختار کلی کنترلر فازی

۴.۱۱ تمرین

۱. عملکرد کنترلر فازی طراحی شده در آزمایشگاه در حالتی که خطای یک سیگنال نویزی است شبیه سازی کنید.(نویز را به صورت نویز سفید در نظر بگیرید).

۲. کنترلر فازی پیشنهاد شده در آزمایش خطای مشتق خطای از ورودی دریافت می کرد. به نظر شما لزوم استفاده از مشتق خطای چیست؟ آیا می توان تنها با فیدبک گرفتن از خطای خودرو توصیف شده در قسمت شرح آزمایش را کنترل کرد؟

۳. کنترلر فازی طراحی شده در آزمایشگاه را به گونه ای تغییر دهید که تنها خطای شیب جاده را از ورودی دریافت کند. هدف کنترلر ثابت نگاه داشتن سرعت خودرو بر روی 50 km/h است.

آزمایش یازدهم(کنترلر PID فازی)

۱.۱۲ پیش گزارش

فرض کنید قصد دارید کنترلر PID را توسط یک کنترلر فازی پیاده سازی کنید.

۱. چه سیگنال هایی را به عنوان ورودی کنترلر فازی در نظر می گیرید.
۲. قوانینی را که به صورت اگر-آنگاه برای پیاده سازی کنترلر در نظر می گیرید را بیان کنید.

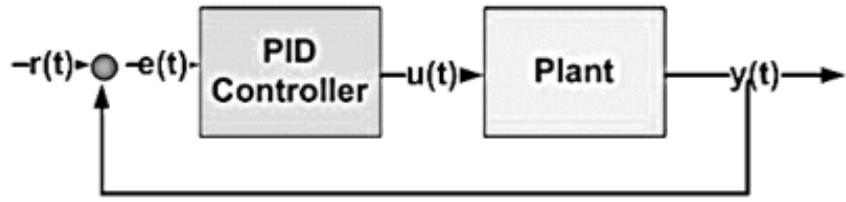
۲.۱۲ مقدمه

در این آزمایش قصد داریم عملکرد کنترلر PID را توسط یک کنترلر فازی شبیه سازی کنیم. در این قسمت ابتدا به مروری کوتاه بر ساختار کنترلر PID پرداخته می شود. سپس ساختار کنترلر فازی برای شبیه سازی عملکرد کنترلر PID بررسی می شود.

۱.۲.۱۲ کنترلر PID

کنترل کننده PID یک کنترل کننده قدرتمند و در عین حال ساده، برای سیستم های کنترلی می باشد که تابع تبدیل و خروجی آن به صورت معادلات ۱.۱۲ است. ساختار کلی کنترلر در شکل ۱.۱۲ نمایش داده شده است. ورودی این کنترلر خطأ و خروجی آن ورودی سامانه است.

$$\begin{aligned} G(s) &= K_p + \frac{K_i}{s} + K_d s \\ u(t) &= K_p[e(t) + \frac{1}{T_i} \int_0^t e(r) dr + T_d \dot{e}] \\ T_i &= \frac{K_p}{K_i} \\ T_d &= \frac{K_d}{K_p} \end{aligned} \quad (1.12)$$



شکل ۱.۱۲: ساختار کلی کنترلر *PID*

کنترلر *PID* فازی

برای طراحی کنترلر *PID* فازی ورودی های کنترلر به صورت خطأ و مشتق خطأ در نظر گرفته می شود. خروجی کنترلر نیز ضرایب $K_{P'}$ و $K_{d'}$ و α که به صوت زیر تعریف می شوند در نظر گرفته می شود. در نهایت بعد از به دست آوردن ضرایب توسط یک بلوک معکوس کننده ضرایب K_p ، K_d و K_i به ضرایب $K_{P'}$ و $K_{d'}$ و α تبدیل می شوند. ساختار ورودی و خروجی کنترلر فازی *PID* در شکل ۲.۱۲ نمایش داده شده است.

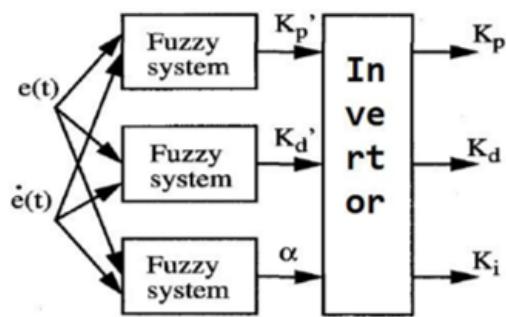
$$K_{P'} = \frac{K_p - K_{pmin}}{K_{pmax} - K_{pmin}}$$

$$K_{d'} = \frac{K_d - K_{dmin}}{K_{dmax} - K_{dmin}}$$

$$T_i = \alpha T_d \quad (2.12)$$

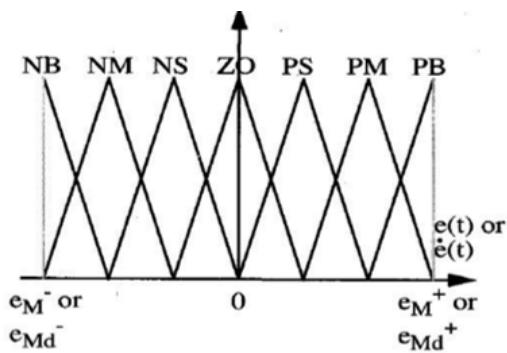
$$T_i = \alpha T_d$$

توابع عضویت مختلفی را می توان برای ورودی های کنترلر *PID* فازی در نظر گرفت. یک نمونه از توابع عضویت که می

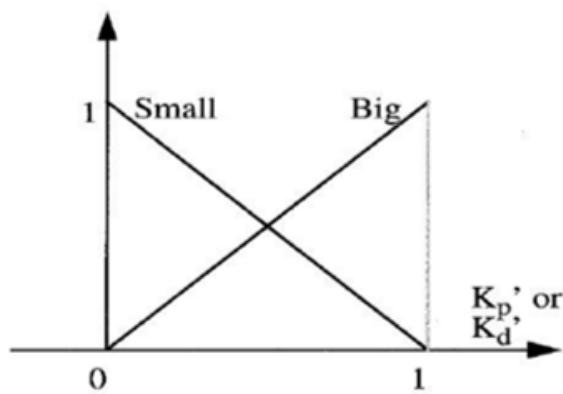


شکل ۲.۱۲: رابطه بین ورودی و خروجی های کنترلر *PID* فازی

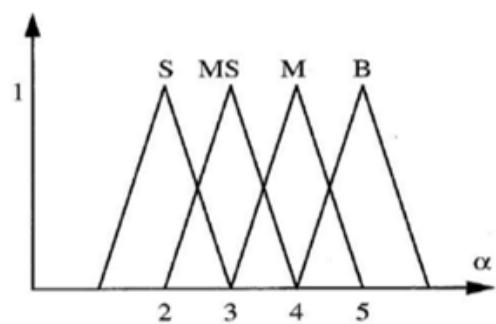
توان برای خطأ و مشتق خطأ در نظر گرفت در شکل ۳.۱۲ نمایش داده شده است. برای $K_{P'}$ ، $K_{d'}$ و α نیز انتخاب های متفاوتی برای توابع عضویت وجود دارد. یکی نمونه از توابع عضویت برای $K_{P'}$ در شکل ۴.۱۲ و یک نمونه تابع عضویت برای α در شکل نمایش داد شده است. بعد از مشخص شدن توابع عضویت برای متغیرهای مختلف سیستم فازی برای کامل شدن ساختار باید به تعریف قوانین پرداخته شود. قوانین سیستم فازی را می توان به گونه های مختلفی تعریف کرد. یک نمونه از قوانین برای $K_{P'}$ ، $K_{d'}$ و α به ترتیب در شکل های ۶.۱۲، ۷.۱۲ و ۸.۱۲ نمایش داده شده است. با تعریف شدن



شکل ۳.۱۲: تابع عضویت خطأ و مشتق خطأ



شکل ۴.۱۲: تابع عضویت گین ها



شکل ۵.۱۲: تابع عضویت α

قوانين و مشخص شدن توابع عضویت با انتخاب یک فازی ساز و یک ضد فازی ساز^۱ طراحی کنترلر فازی PID پایان می پذیرد. برای انتخاب فازی ساز و ضد فازی ساز نیز گزینه های مختلفی وجود دارد برای مثال می توان از فازی ساز singleton و ضد فازی ساز (با در نظر گرفتن product – inference – engine) استفاده کرد.

^۱ defuzzifier

		$\dot{e}(t)$						
		NB	NM	NS	ZO	PS	PM	PB
e(t)	NB	B	B	B	B	B	B	B
	NM	S	B	B	B	B	B	S
	NS	S	S	B	B	B	S	S
	ZO	S	S	S	B	S	S	S
	PS	S	S	B	B	B	S	S
	PM	S	B	B	B	B	B	S
	PB	B	B	B	B	B	B	B

شکل ۶.۱۲: قوانین $K_{P'}$

		$\dot{e}(t)$						
		NB	NM	NS	ZO	PS	PM	PB
e(t)	NB	S	S	S	S	S	S	S
	NM	B	B	S	S	S	B	B
	NS	B	B	B	S	B	B	B
	ZO	B	B	B	B	B	B	B
	PS	B	B	B	S	B	B	B
	PM	B	B	S	S	S	B	B
	PB	S	S	S	S	S	S	S

شکل ۷.۱۲: قوانین $K_{d'}$

		$\dot{e}(t)$						
		NB	NM	NS	ZO	PS	PM	PB
e(t)	NB	2	2	2	2	2	2	2
	NM	3	3	2	2	2	3	3
	NS	4	3	3	2	3	3	4
	ZO	5	4	3	3	3	4	5
	PS	4	3	3	2	3	3	4
	PM	3	3	2	2	2	3	3
	PB	2	2	2	2	2	2	2

شکل ۸.۱۲: قوانین a

۳.۱۲ شرح آزمایش

هدف از این آزمایش، طراحی و پیاده‌سازی یک سیستم فازی PID برای هدایت ربات از موقعیت اولیه به موقعیت دلخواه است. ورودی‌های سیستم فازی، مکان هدف نسبت به مکان کنونی ربات و خروجی آن فرمان کنترلی موتورها می‌باشد. با

انتخاب قوانین مناسب، کنترل صورت خواهد گرفت.

برای پیاده سازی کنترلر فازی مراحل زیر را دنبال کنید:

۱. با توجه به شکل های ۳.۱۲ تا ۵.۱۲ (با استفاده از جعبه ابزار فازی متلب) توابع عضویت متغیر های ورودی و خروجی را مشخص کنید(خطا و مشتق خطا به عنوان ورودی سیستم و $K_{P'}$ ، $K_{d'}$ و α به عنوان خروجی سیستم در نظر گرفته می شود)
۲. با استفاده از قسمت انتخاب قوانین در جعبه ابزار متلب قوانین سیستم را مطابق با شکل های ۶.۱۲ ، ۷.۱۲ و ۸.۱۲ تعریف کنید.
۳. فازی ساز را به صورت *singleton* و غیر فازی ساز را به صورت *center – average* در نظر بگیرید.
۴. با توجه به معادلات ۲.۱۲ و با در نظر گرفتن K_p در محدوده یک بلوک در سیمولینک طراحی کنید که K_p ، $K_{P'}$ و α را از ورودی دریافت کند و K_i و T_i را به عنوان خروجی باز گرداند.
۵. خروجی های سیستم فازی را به عنوان ورودی به بلوک طراحی شده در مرحله ۳ دهید.

بعد از طراحی کردن کنترلر *PID* فازی کنترلر را بر روی سامانه توصیف شده در معادلات ۳.۱۲ و با در نظر گرفتن $\sin(t)$ به عنوان ورودی مرجع امتحان کنید و نمودار خطای خروجی (تفاوت بین خروجی سامانه و ورودی مرجع) را رسم کنید.

$$H(s) = \frac{40}{s + 1} \quad (3.12)$$

۴.۱۲ تمرین

۱. با فرض وجود نویز در خروجی سامانه عملکرد کنترلر فازی طراحی شده را شبیه سازی و بررسی کنید.
۲. تعداد قوانین کنترلر فازی را افزایش دهید و تاثیر آن را در عملکرد کنترلر بررسی کنید.

آزمایش دوازدهم ($c - mean$)

۱.۱۳ پیش گزارش

۱. ضعف های الگوریتم K میانگین را بیان کنید. این ضعف ها چگونه توسط الگوریتم $c - mean$ فازی بهبود می یابد. و $k - mean$ را بیان کنید.

۲.۱۳ مقدمه

در آزمایش های قبل با الگوریتم $k - mean$ برای دسته بنده داده ها آشنا شدید. ممکن است مرز مشخصی برای دسته های داده ها وجود نداشته باشد. در چنین شرایطی کارآیی الگوریتم $k - mean$ کاهش می یابد. الگوریتم $c - mean$ فازی مشابه الگوریتم $k - mean$ است با این تفاوت که خروجی این الگوریتم احتمال متعلق بودن داده ها به دسته های مشخص شده است.

تابع هزینه در نظر گرفته شده برای الگوریتم $c - mean$ به صورت معمله ۱.۱۳ است. در معادله زیر n مشخص کننده تعداد داده ها و i مشخص کننده تعداد دسته ها و V_i مشخص کننده مرکز دسته i است. هدف الگوریتم $c - mean$ کمینه کردن تابع هزینه در نظر گرفته شده در معادله ۱.۱۳ است. این تابع هزینه تلاش می کند تراکم داده ها درون یک دسته را افزایش دهد با زیاد شدن تراکم داده ها درون یک دسته هم زمان تراکم داده ها در خارج از دسته ها کاهش می یابد.

$$\sum_{k=1}^n \sum_{c=1}^i \mu_{ik} \| x_{ik} - v_i \| \quad (1.13)$$

در شکل ۱.۱۳ مراحل الگوریتم $c - mean$ نمایش داده شده است. در این آزمایش قصد داریم به پیاده سازی الگوریتم $c - mean$ فازی با استفاده از ابزار متلب بپردازیم.

- Suppose there are n data points, $X = \{x_1, \dots, x_n\}$. Fix c , $2 \leq c < n$, and initialize $U^{(0)} \in M_c$.
- At iteration I , $I = 0, 1, 2, \dots$ compute the c -mean vectors

$$v_i^{(I)} = \frac{\sum_{k=1}^n \mu_{ik}^{(I)} x_k}{\sum_{k=1}^n \mu_{ik}^{(I)}}$$

where $[\mu_{ik}^{(I)}] = U^{(I)}$, and $i = 1, 2, \dots, c$.
- Update $U^{(I)}$ to $U^{(I+1)}$ using

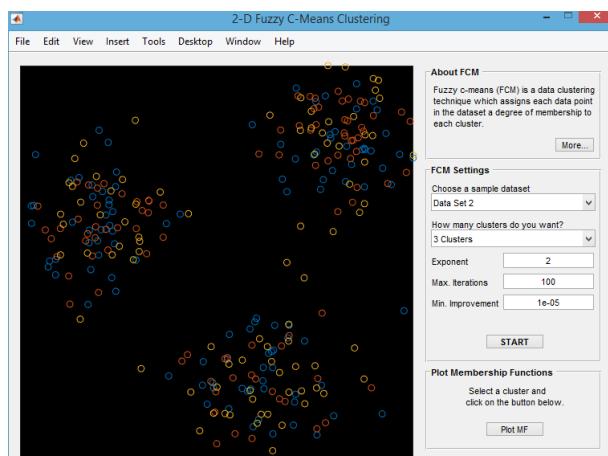
$$\mu_{ik}^{(I+1)} = \begin{cases} 1 & \|x_k - v_i^{(I)}\| = \min_{1 \leq j \leq c} (\|x_k - v_j^{(I)}\|) \\ 0 & \text{otherwise} \end{cases}$$
- Compare $U^{(I)}$ with $U^{(I+1)}$: if $\|U^{(I+1)} - U^{(I)}\| < \epsilon$ for a small constant ϵ stop; otherwise, set $I = I + 1$ and go to Step 2.

شکل ۱.۱۳: مراحل الگوریتم $c - mean$

۱.۲.۱۳ آشنایی با جعبه ابزار $c - mean$ فازی متلب

۳۱- برای این نوع طبقه بندی، ابزار مخصوصی در متلب تعبیه شده که برای مرکز دسته ها یک فرض اولیه در نظر می گیرد. مرکز دسته ها یعنی موقعیت میانگین هر دسته بندی. از آنجایی که حدس اولیه عموماً اشتباه است، ابزار متلب به هر نقطه،تابع عضویتی نسبت می دهد که قادر است مراکز دسته هارا به محل اصلی آنها هدایت کند. این کار با کمینه کردن تابع هدف رخ می دهد که مشخص کننده فاصله بین هر نقطه با مرکز دسته ای است که توسط تابع عضویت آن نقطه، وزن پیدا کرده است.

با تایپ کردن `fcmdemo` در `command - window` پنچره ای به شکل ۲.۱۳ باز می شود. پنچره باز شده محیط `fcm - setting` (مربع سمت راست بالای پنچره باز شده) می توانید داده ورودی و تعداد دسته های مورد نظر را مشخص کرد. بعد از مشخص کردن تنظیمات با کلیک بر روی `Start` الگوریتم اجرا می شود.



شکل ۲.۱۳: محیط fcm

۲.۲.۱۳ ابزار فازی در محیط $command - line$ متلب

برای استفاده از ابزار فازی می توان از `command - line` متلب نیز استفاده کرد. در `[C, U] = fcm(dataset, N, options)` با استفاده از دستور `fcm` می توان الگوریتم دسته بندی فازی را اجرا کرد. ورودی های این دستور داده های در نظر گرفته و یک بردار `options` است. بردار `options` به صورت `options = [exponent maxIterations, minImprovement, displayObjectiveFunction];` دارد.

المان اول در بردار *options* مشخص کننده بیشینه مجاز تعداد تکرار الگوریتم است . المان دوم در بردار *options* مشخص کننده کمترین تغییرات مراکز دسته ها در هر تکرار نسبت به تکرار قبل (برای مشخص شدن همگرا شدن و یا نشدن الگوریتم) است. در صورت *True* بودن المان سوم در بردار *options* مقدار تابع هزینه بعد از هر تکرار نمایش داده می شود و در غیر اینصورت این مقدار نمایش داده نمی شود. خروجی های دستور *fcm* مراکز داده ها و ماتریس احتمال متعلق بودن هر داده به هر دسته است.

۳.۱۳ شرح آزمایش

در این آزمایش ابتدا قصد داریم با استفاده از جعبه ابزار دسته بندی فازی متلب به دسته بندی داده ها بپردازیم سپس با استفاده از *command-line* به دسته بندی داده های چند بعدی پرداخته می شود.

۱.۳.۱۳ دسته بندی با استفاده از جعبه ابزار *fcm*

در این قسمت قصد داریم به دسته بندی داده های ۲ بعدی با استفاده از جعبه ابزار *fcm* بپردازیم.
مراحل زیر را دنبال کنید:

۱. در *Command – Windows* *fcmdemo* متلب دستور *fcmdemo* را تایپ کنید.
۲. نمونه مجموعه داده‌ی شماره ۵ را از منوی *Choose a sample dataset* انتخاب کنید.
۳. در منوی بعدی، تعداد دسته‌ها را برابر با ۵ دسته قرار دهید.
۴. *START* را انتخاب کنید و نتیجه را مشاهده نمایید.
همانطور که مشاهده می‌شود، حدس مرکز اولیه مرحله به مرحله توسط نرم افزار مشخص شده و به مرکز بهینه نزدیک تر می‌گردد.
۵. یکی از دسته‌ها را انتخاب کنید. با کلیک بر روی *PLOTFM* سطح تابع عضویت آن را مشاهده کنید.

۲.۳.۱۳ دسته بندی *fcm* در *Command – line*

فرض کنید یک شرکت تبلیغاتی شبکه اجتماعی کاریابی، برای گسترش عملکرد شبکه‌ی خود تصمیم دارد کاربران خود را با داشتن اطلاعات زیر به ۷ دسته تقسیم کند. بدین صورت که با توجه به اطلاعات تقسیم بندی شده، تبلیغات منحصر به فردی برای گروه‌های مختلف ارسال شود. برای مثال، گروه شامل افراد جوان تر، تبلیغات مربوط به فناوری‌های روز شبکه را دریافت می‌کنند. درحالی که افراد مسن، تبلیغاتی درباره ساده سازی شبکه و کاربری کم دردسرتر مشاهده خواهند کرد. داده شرکت در اختیارتان قرار داده می شود. قصد داریم با استفاده از *fcm* داده های شرکت را به ۷ دسته مناسب تقسیم

کنیم.

برای دسته بندی داده ها مراحل زیر را دنبال کنید:

۱. در متلب به آدرس *CILab* مراجعه کنید. مجموعه داده *SocialNetworkAds.csv* را انتخاب کنید تا اطلاعات را وارد متلب کنید.
۲. داده های مورد نیاز را انتخاب کرده و بر روی *ImportSelection* قرار دهید و بر روی *Numeric – Matrix* را برابر با *Output – Type* کلیک کنید تا اطلاعات در یک متغیر در محیط متلب ذخیره شوند.
۳. بر روی متغیر ایجاد شده در *Workspace* راست کلیک کرده و گزینه *Save – As* را انتخاب کنید. حال مجموعه داده را با نام *SNdataset.m* ذخیره کنید.
۴. فایل ذخیره شده را باز کنید و در انتهای آن دستور زیر را اضافه کنید.

$$\text{numberOfClusters} = 7;$$

$$[\text{center}, U] = \text{fcm}(\text{SocialNetworkAds}, \text{numberOfClusters});$$

با استفاده از این دستور، مشاهده می شود که تمام مراحل تقسیم بندی اجرا شده است. در نهایت متغیرهای U_{center} در *Workplace* ذخیره می شوند.

۴.۱۳ تمرین

۱. مراحل دسته بندی انجام شده در بخش آشنایی با جعبه ابزار $c - mean$ فازی متلب را برای مجموعه داده سوم به جای مجموعه داده پنجم(از مجموعه داده آماده متلب در بخش انتخاب مجموعه داده) تکرار کنید.
۲. مراحل دسته بندی انجام شده در بخش آشنایی با جعبه ابزار $c - mean$ فازی متلب را با دیگر برای مجموعه داده پنجم(مجموعه داده آزمایش شده در آزمایشگاه) تکرار کنید آیا خروجی به دست آماده با خروجی به دست آماده در آزمایشگاه در قسمت دسته بندی توسط جعبه ابزار *fcm* یکسان است؟ پاسخ خود را توجیه کنید.

آزمایش سیزدهم (*anfis*)

۱.۱۴ پیش گزارش

۱. یک شبکه مزایایی نسبت به شبکه عصبی و فازی دارد؟

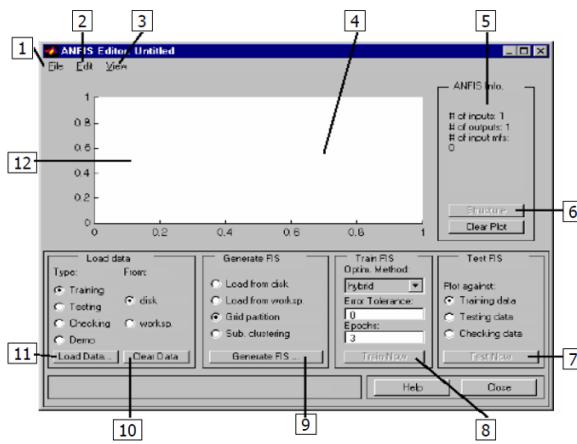
۲. کاربردهای شبکه *anfis* را بیان کنید

۲.۱۴ مقدمه

سیستم عصبی فازی (*ANFIS(Adaptive – Neural – Fuzzy – Inference – System)*) در واقع ترکیب یک سیستم فازی با یک شبکه عصبی است. در شبکه فازی تعین توابع عضویت، ویژگی آن ها و ... توسط دانش انسانی صورت می گرفت. در سیستم عصبی-فازی از یک شبکه عصبی برای انتخاب پارامترهای سیستم فازی استفاده می گردد. در این آزمایش قصد داریم با استفاده از جعبه ابزار *anfis* مطلب به پیاده سازی یک سیستم *anfis* بپردازیم. در ابتدا به چگونگی استفاده از جعبه ابزار *anfis* مطلب پرداخته می شود.

۱.۲.۱۴ جعبه ابزار *anfis* مطلب

با تایپ کردن *anfisedit* در *command – line* مطلب پنجه ای به شکل ۱.۱۴ نمایش داده می شود. در مربع شماره ۱ می توانید سیستم *anfis* ایجاد شده را ذخیره کنید و یا یک سیستم *anfis* جدید ایجاد کنید. در مربع شماره ۲ و با کلیک بر روی *Edit* می توانید پارامترهای سیستم فازی را مشخص کنید. در صورت در اختیار داشتن یک فایل *.fis* می توانید در قسمت مشخص شده توسط مربع ۳ آن را بارگزاری کنید. مربع شماره ۴ محل رسم را نمایش می دهد. در مربع شماره ۵ مشخصات سیستم *anfis* مانند تعداد ورودی ها و خرجی ها نمایش داده می شود. بعد از ایجاد یا باز کردن یک فایل *FIS* توسط گزینه نمایش داده شده با عدد ۶ می توان ساختار شبکه فازی را مشاهده نمود با کلیک کردن بر روی مربع ۷ می توانید داده های تست را رسم کنید. بعد از مشخص کردن مشخصات سیستم فازی با انتخاب کردن مربع ۸ می توان شبکه را آموزش داد. در مربع شماره ۹ می توانید یک فایل *Fis* را به روش های مختلف تولید کنید. در مربع شماره ۱۰ می توانید



شکل ۱.۱۴: جعبه ابزار *anfistool*

داده های بار گزاری شده را پاک کنید. در مربع شماره ۱۱ می توانید داده های ورودی و خروجی را بار گزاری کنید. برای رسم نمودار در ناحیه مشخص شده توسط عدد ۱۲ داده های تست با $\cos(t)$ آموزش داده می شود.

۳.۱۴ شرح آزمایش

در این آزمایش قصد داریم توسط یک شبکه *anfis* به تخمین تابع $\cos(t)$ بپردازیم. برای تخمین تابع $\cos(t)$ توسط شبکه *anfis* مراحل زیر را دنبال کنید.

۱. داده های ورودی و خروجی تابع $\cos(t)$ را در *workspace* ذخیره کنید.
۲. در *command-line* مطلب تایپ کنید تا جعبه ابزار *anfis* نمایش داده شود.
۳. در قسمت *loaddata* در جعبه ابزار *anfis* داده های ذخیره شده در *workspace* را در محیط *anfis* وارد کنید.
۴. گزینه *Generate - Fis* را انتخاب کنید و مشخصات سیستم فازی و تعداد توابع عضویت را وارد کنید.
۵. با انتخاب گزینه *Train - Fis* عملیات یادگیری شبکه *Fis* را شروع کنید و نمودار خطای را مشاهده کنید.

۴.۱۴ تمرین

۱. سعی کنید تابع $\cos^2(t) + \sin(t)$ را توسط یک شبکه *anfis* تخمین بزنید.

پروژه بخش فازی

۱.۱۵ پیش گزارش

۱. با مراجعه به اینترنت معادلات دینامیک مربوط به خودرو بدون سرنشین چهار چرخ را بیان کنید.
۲. با مراجعه به اینترنت معادلات دینامیک مربوط به ربات نقاش ۲ بعدی را بیان کنید.

۲.۱۵ مقدمه

در آزمایش های گذشته به طراحی کنترلر توسط سیستم فازی پرداخته شد. در این آزمایش قصد داریم به پیاده سازی سخت افزاری الگوریتم های بیان شده بپردازیم. برای پیاده سازی سخت افزاری از خودرو زمینی چهار چرخ بدون سرنشین^۱ و یا ربات نقاش^۲ بعدی استفاده می شود. در ادامه به معرفی کوتاه ربات نقاش^۲ بعدی و خودرو زمینی بدون سرنشین پرداخته می شود.

۱.۲.۱۵ خودرو زمینی بدون سرنشین

خودرو زمینی بدون سرنشین یک ربات^۴ چرخ با مکانیزم حرکت تانکی و خودکار بوده که قابلیت طی مسیر به صورت خودکار و ارتباط با عامل های دیگر از طریق بی سیم را نیز دارد. این ربات مجهز به حلقه^۶ تایی سونار برای مانع یابی و همچنین^۲ عدد انکوادر جهت مکان یابی بوده و قابلیت نصب سنسورهای مختلفی در آن در نظر گرفته شده است. توضیحات مفصل این ربات در (پیوست ۱) انجام شده است. (حتماً پیش از آزمایش، پیوست مطالعه شود). تصویر این ربات در شکل زیر نمایش داده شده است.

¹ UGV(unmanned – ground – veichle)



شکل ۱.۱۵: خودروی زمینی بدون سرنویس

۲.۲.۱۵ ربات نقاش ۲ بعدی

ربات نقاش یک ربات دو بعدی با دو درجه آزادی ، همانطور که در شکل زیر مشاهده می شود، این ربات از چهار لینک (بازو) که از یک طرف به دو محرک (موتور) DC و از طرف دیگر به یکدیگر متصل شده اند تشکیل شده است. نقطه اتصال لینک ها به هم سر موثر نمایدند می شود که با اعمال فرمان مناسب به موتورها ، سر موثر به نقاط مطلوب در صفحه $y - x$ حرکت داده می شود. سر موثر شامل یک قلم میباشد که این قلم جهت بر جای گذاشتن مسیر حرکت سر موثر به کار رفته است. سر موثر ربات به یک



شکل ۲.۱۵: ربات نقاش ۲ بعدی

۳.۱۵ شرح آزمایش

در این بخش ۲ آزمایش برای پیاده سازی بر روی ربات نقاش ۲ بعدی و خودرو بدون سرنویس در نظر گرفته شده است . از بین این ۲ آزمایش ۱ آزمایش را انتخاب کرده و به پیاده سازی سخت افزاری آن بپردازید.

۱.۳.۱۵ آزمایش ۱ - کنترل خودروی زمینی بدون سرنشین

در این قسمت به پیاده سازی سخت افزاری کنترل *PID* فازی شبیه سازی شده، بر روی خودرو زمینی بدون سرنشین پرداخته می شود.

قصد داریم خودرو بدون سرنشین را به گونه ای کنترل کنیم که بر روی خط $x = 10$ (یا $y = 10$) حرکت کند. برای کنترل خودروی بدون سرنشین مراحل زیر را دنبال کنید:

۱. ابتدا از صحت کنترلر پیاده سازی شده در آزمایش ۱۱ اطمینان حاصل کنید.

مدل ساده شده خودرو بدون سرنشین به صورت رابطه ۱.۱۵ است. ابتدا کنترلر طراحی شده در آزمایش ۱۱ را برروی این مدل شبیه سازی کنید. ورودی مرجع را برابر با یک عدد ثابت در نظر بگیرید. در صورت عملکرد صحیح کنترلر خطای بین خروجی سامانه و ورودی مرجع به سمت صفر میل می کند.

$$H(s) = \frac{40}{0.02s^2 + s} \quad (1.15)$$

۲. فایل سیمولینک مربوط به کنترل خودرو بدون سرنشین در اختیارتان قرار داده می شود. فایل را در محیط سیمولینک باز کنید.

۳. ورودی مرجع کنترلر را برابر با ۱۰ در نظر بگیرید. مقدار pwm_1 (pwm_2) را برابر با خروجی کنترلر فازی قرار دهید. موقعیت خودرو بدون سرنشین را نیز به عنوان ورودی کنترلر در نظر بگیرید. خطای بین ورودی مرجع و موقعیت خودرو بدون سرنشین را رسم کنید.

۲.۳.۱۵ آزمایش ۲ - شناسایی ربات نقاش ۲ بعدی

در این قسمت قصد داریم با استفاده از یک شبکه *anfis* به شناسایی ربات نقاش ۲ بعدی بپردازیم. برای شناسایی ربات نقاش ۲ بعدی مراحل زیر را دنبال کنید:

۱. ابتدا فایل ربات را که در اختیارتان قرار گرفته است را اجرا کنید. با اجرا کردن این فایل ربات شروع به نقاشی یک دایره خواهد کرد.

۲. این ربات ۲ سیگنال را از ورودی دریافت می کند (سیگنال های u_1, u_2 در فایل سیمولینک). این ۲ سیگنال را به عنوان ورودی در *workspace* متلب ذخیره کنید و زاویه بازو ها را به عنوان خروجی در *workspace* ذخیره کنید.

۳. داده های ورودی و خروجی را به شبکه *anfis* طراحی شده در آزمایش ۱۳ دهید و شبکه را شناسایی کنید.

۴. خروجی شبکه *anfis* را به ازای u_1, u_2 برابر با ۴ تخمین بزنید.

۵. در فایل سیمولینک مقادیر u_1, u_2 را برابر با ۴ قرار دهید و برنامه را بر روی ربات اجرا کنید.

۶. مقدار زاویه تخمین زده شده توسط شبکه *anfis* را با مقدار زاویه بازوهای ربات نقاش ۲ بعدی مقایسه کنید.

۱. خروجی های حاصل شده از بخش عملی را بخشن تئوری مقایسه کنید. ضعف های الگوریتم های استفاده شده در پیاده سازی سخت افزاری چیست؟

پیوست

آ ربات نقاش ۲ بعدی

ربات نقاش دو بعدی در واقع یک ربات با دو بازو است که بازوها از یک طرف به یکدیگر متصل شده اند و نقش گیرنده قلم یا هر وسیله ای که قرار است با آن نقاشی رسم شود را دارند. هر یک از بازوهای این ربات دارای دو درجه آزادی است که امکان حرکت آزادانه را به سادگی به آن می دهد.

برای حرکت کامل در صفحه ۲ بعدی وجود ۳ درجه آزادی کافیست. یعنی تنها کافیست با دو درجه آزادی در مختصات مطلوب قرار گرفت و با درجه آزادی سوم با زاویه (Orientation) دلخواه در آن نقطه قرار گرفت. وجود ۴ درجه آزادی در این ربات باعث می شود با تعداد حالت های بیشتری برای مفاصل (بازوها) بتوان در موقعیت مطلوب قرار گرفت. وجود درجه آزادی چهارم در این ربات اصطلاحاً باعث ایجاد Redundancy در آن می شود. وجود درجات آزادی بالاتر از حد نیاز در ربات ها بخصوص برای مقابله با موانع کاربرد دارد.

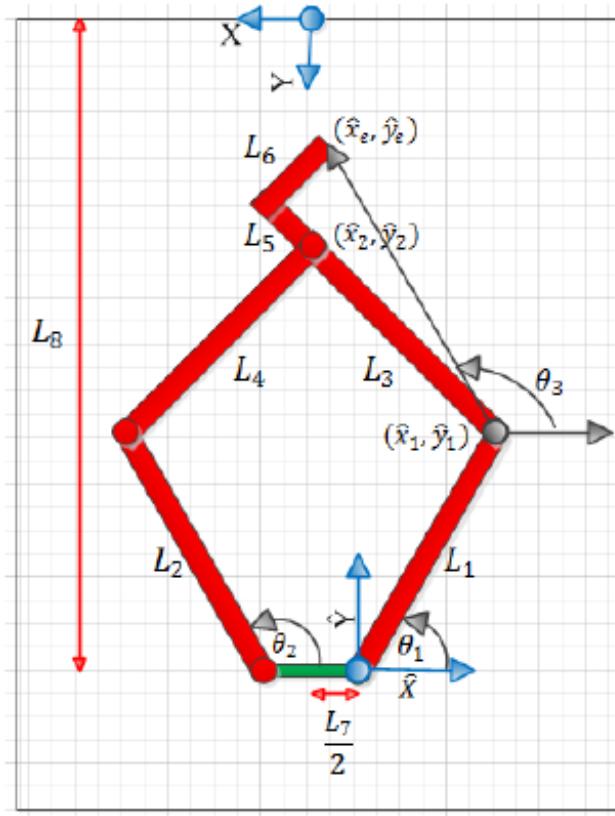
با توضیحات داده شده به نظر شما برای حرکت کامل در فضای ۳ بعدی حداقل چند درجه آزادی لازم است؟ اعمال نیرو به بازوهای ربات نقاش از طریق دو عدد سرورو موتور در مفصل دو بازو انجام می شود. همچنین برای فیدبک گرفتن از محل سر بازوها که به هم متصل است از انکوادر بر روی موتورها استفاده می شود. به عبارتی دیگر با دانستن محل موتورها می توان محل سر بازوها را تنظیم نمود.

جهت کار با ربات باید با داشتن مختصات مطلوب قلم، موقعیت لازم هر یک از بازو ها و سپس موتور ها را بدست آوریم. این مسئله در رباتیک، مسئله سینماتیک معکوس نامیده می شود. برای کشیدن هر طرحی با ربات لازم است از معادلات سینماتیک معکوس ربات استفاده کرده و موقعیت لازم برای هر یک از بازو ها و موتور ها را بدست آورد و موتور ها و بازو ها را به آن موقعیت هدایت کرد تا نقطه انتهایی ربات (End Effector) که در این ربات همان قلم است، در موقعیت مطلوب قرار بگیرد.

مسئله سینماتیک مستقیم در ربات ها، عکس مطلب فوق می باشد. یعنی در سینماتیک مستقیم ربات، با فرض داشتن موقعیت موتور ها و بازو ها، هدف بدست آوردن نقطه انتهایی ربات (قلم) می باشد.

معمولًا مسئله سینماتیک معکوس در ربات ها، مسئله پیچیده تری است تا مسئله سینماتیک مستقیم؛ چرا که در

سینماتیک معکوس معمولاً باید به حل دستگاهی از معادلات غیر خطی (و غالباً شامل عبارت‌های مثلثاتی) بپردازیم. این گونه معادلات اکثراً پاسخی به فرم بسته ندارند و برای حل آن‌ها باید به روش‌های عددی پرداخت. اکنون معادلات مربوط به مسئله سینماتیک مستقیم و سینماتیک معکوس ربات نقاش دو بعدی را بررسی می‌کنیم. کدام معادلات مربوط به سینماتیک مستقیم هستند؟ کدام مربوط به سینماتیک معکوس؟ چرا؟



$$\hat{x}_1 = L_1 \cos \theta_1$$

$$\hat{y}_1 = L_1 \sin \theta_1$$

$$\hat{x}_2 = \hat{x}_e - \sqrt{L_5^2 + L_6^2} \cos \left(\theta_3 - \tan^{-1} \left(\frac{L_6}{L_5} \right) + \tan^{-1} \left(\frac{L_6}{L_3 + L_5} \right) \right)$$

$$\hat{y}_2 = \hat{y}_e - \sqrt{L_5^2 + L_6^2} \sin \left(\theta_3 - \tan^{-1} \left(\frac{L_6}{L_5} \right) + \tan^{-1} \left(\frac{L_6}{L_3 + L_5} \right) \right)$$

$$\theta_1 = \cos^{-1} \left(\frac{\hat{x}_e}{\sqrt{\hat{x}_e^2 + \hat{y}_e^2}} \right) - \cos^{-1} \left(\frac{L_1^2 + \hat{x}_e^2 + \hat{y}_e^2 - (L_3 + L_5)^2}{2L_1\sqrt{\hat{x}_e^2 + \hat{y}_e^2}} \right)$$

$$\theta_2 = \cos^{-1} \left(\frac{\hat{x}_2 + L_6}{\sqrt{(\hat{x}_2 + L_6)^2 + \hat{y}_2^2}} \right) + \cos^{-1} \left(\frac{L_2^2 + (\hat{x}_2 + L_6)^2 + \hat{y}_2^2 - L_4^2}{2L_2\sqrt{(\hat{x}_2 + L_6)^2 + \hat{y}_2^2}} \right)$$

$$\theta_3 = \cos^{-1} \left(\frac{\hat{x}_e - \hat{x}_1}{\sqrt{(\hat{x}_e - \hat{x}_1)^2 + (\hat{y}_e - \hat{y}_1)^2}} \right)$$

آ. ترسیم با ربات های نقاش

هدف نهایی ربات های نقاش، ترسیم طرح های دلخواه می باشد. یکی از راه های ترسیم طرح با این ربات ها داشتن معادلات طرح مورد نظر می باشد. با داشتن این معادلات می توان به کمک معادلات سینماتیک معکوس ربات، طرح مورد نظر را رسم کرد. بطور مثال برای ترسیم یک دایره کافیست هر یک از بازو ها یک حرکت سینوسی با دامنه و فرکانس یکسان و اختلاف فاز ۹۰ درجه نسبت به هم داشته باشند. اما واضح است که برای اکثر تصاویر موجود (تصاویر گرفته شده با دوربین و ...) بدست آوردن این گونه معادلات بسیار پیچیده و عملاً غیر ممکن است. در این گونه موارد می توان با استفاده از الگوریتم های مرسوم پردازش تصویر، لبه ها و کانتور های تصاویر را یافت و باز هم به کمک معادلات سینماتیک معکوس، به ترسیم این نقاط پرداخت. باید این نکته را هم در نظر گرفت که اکثر ربات های نقاش موجود تنها از یک رنگ استفاده می کنند. به علاوه این که شدت این رنگ نیز معمولاً قابل تنظیم نیست. در نتیجه تصویر رسم شده با این ربات ها معمولاً تنها از یک رنگ بجز پس زمینه تشکیل شده است. در نتیجه این گونه ربات ها برای ترسیم تصاویری با جزئیات رنگی بالا مناسب نمی باشند. برای وضوح بیشتر بهتر است تصویر ورودی به گونه ای پردازش شود تا ضمن حفظ کیفیت و جزئیات تنها از دو رنگ (معمولًا سفید و غیرسفید) تشکیل شده باشد. بدین منظور می توان از الگوریتم هایی نظیر Adaptive Thresholding در پردازش تصویر استفاده کرد. علاقمندان می توانند از توابعی همچون cv2.adaptiveThreshold ، cv2.threshold ... در ماژول OpenCV و cv2.findContours استفاده کنند.

در شکل ؟؟ نمونه ای از تصویر ورودی، و خروجی ترسیم شده توسط ربات های نقاش را می بینید.



(ب) تصویر ترسیم شده



(آ) تصویر ورودی

شکل ۱.۱۶: نمونه ای از ورودی و خروجی های ربات نقاش

ب ربات زمینی (UGV)

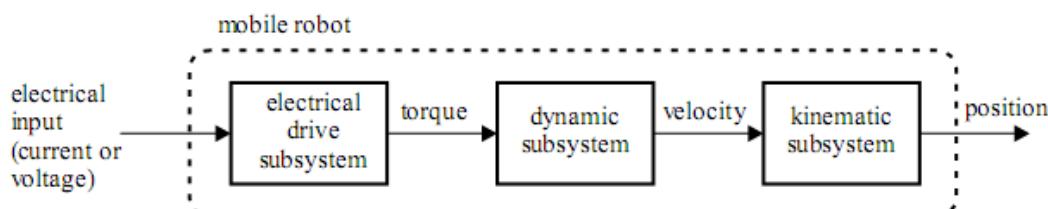
بات زمینی طراحی شده برای این آزمایشگاه یک ربات ۴ چرخ با مکانیزم حرکت تانکی و خودکار بوده که قابلیت طی مسیر به صورت خودکار و ارتباط با عامل‌های دیگر از طریق بی‌سیم را نیز دارد. این ربات مجهز به حلقه ۶ تایی سونار برای مانع‌یابی و همچنین ۲ عدد انکوادر جهت مکان‌یابی بوده و قابلیت نصب سنسورهای مختلفی در آن در نظر گرفته شده است. هرچند ابزارهای فازی و عصبی، امکان مدل‌سازی و طراحی کنترل کننده را به گونه‌ای فراهم می‌آورند که دیگر نیازی به دانستن مدل ریاضی ربات وجود ندارد، با این حال، برای اطمینان از مدل‌سازی و کنترل خودکار این ربات، در ابتدا مدل دینامیک آن تشریح می‌شود. در ادامه برای موقع اضطراری، آموزش نحوه کنترل روبات به صورت دستی و بدون برنامه نویسی توضیح داده شده. لطفاً برای حفظ ایمنی خود، ربات و دیگران، در صورت مشاهده هرگونه رفتار پیش‌بینی نشده یا خطرناک در برنامه‌ی حرکت خودکار، به سرعت روبات را خاموش کرده و با راهنمایی استاد خود و مطالب بخش دوم این پیوست، ربات را به صورت دستی به موقعیت امن منتقل نمایید.



شکل ۲.۱۶: ربات زمینی

ب.۱ مدل‌سازی

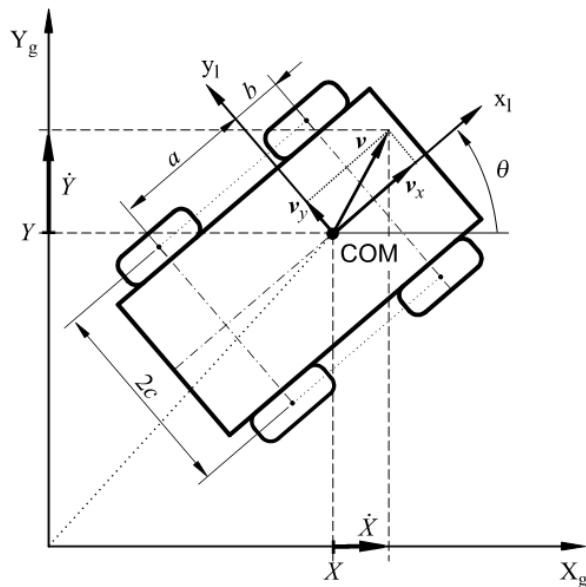
برای کنترل هر ربات بر روی آن یک کامپیوتر محلی تعییه شده است که نرم‌افزارهای مسیریابی و پردازش تصویر و همچنین کنترل رفتاری ربات را بر عهده دارد. برای مدل‌سازی، ربات را به سه بخش تقسیم می‌کنیم که شامل سینماتیک، دینامیک و بخش درایور آن است



شکل ۳.۱۶: سه بخش یک ربات زمینی شامل سینماتیک، دینامیک و درایور

برای درنظر گرفتن مدل سینماتیک ربات، یک دستگاه مختصات مرجع در نظر می‌گیریم و آن را با سه تایی (X_g, Y_g, Z_g) نمایش می‌دهیم. یک دستگاه مختصات محلی هم که مرکز آن روی مرکز جرم ربات هست، را با (x_l, y_l, z_l) نمایش می‌دهیم. با فرض آنکه ربات روی صفحه با سرعت خطی $(v = [v_x, v_y, 0])$ بیان شده در دستگاه محلی، حرکت می‌کند و اینکه بردار مختصات ربات را در دستگاه مرجع بیان می‌کند، می‌توانیم بردار $\dot{q} = [\dot{X}, \dot{Y}, \dot{\theta}]^T$ را بردار سرعت‌های تعمیم یافته بداییم. با توجه به شکل زیر می‌توان، رابطه بین (\dot{x}, \dot{y}) را با (v_x, v_y) به صورت زیر بیان نماییم.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$



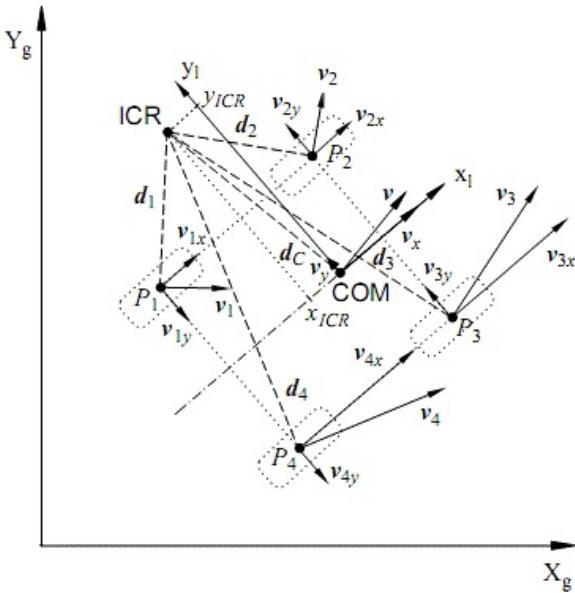
شکل ۴.۱۶: دیاگرام جسم آزاد

این ربات، برخلاف ربات‌هایی با محور دیفرانسیلی، می‌توانند سرعتی در راستای عمود بر صفحه چرخ داشته باشند، و این تنها به دلیل لغزشی است که دارند، بنابراین فرض عدم لغزش برای این ربات‌ها نامعقول می‌باشد. اگر فاصله مرکز چرخش لحظه‌ای را با $d_C = [d_{Cx}, d_{Cy}]^T$ نمایش دهیم و فاصله چرخ (i) ام را از مرکز چرخش لحظه‌ای را با $d_i = [d_{ix}, d_{iy}]^T$ نمایش دهیم، در این صورت می‌توان روابط زیر را بنویسیم:

$$\frac{\|v_i\|}{\|d_i\|} = \frac{\|v\|}{\|d_C\|} = |w|$$

با تعریف نمودن مختصات مرکز چرخش لحظه‌ای در دستگاه محلی به صورت $ICR = (x_{ICR}, y_{ICR}) = (-d_{xC}, -d_{yC})$ خواهیم داشت:

$$\frac{v_x}{y_{ICR}} = -\frac{v_y}{x_{ICR}} = w$$



شکل ۵.۱۶: محورهای مختصات روی قسمت‌های مختلف ربات

بعد از جاگذاری نمودن روابط برای مولفه‌های سرعت هر چرخ می‌توانیم بنویسیم:

$$v_L = v_{1x} = v_{2x}$$

$$v_R = v_{3x} = v_{4x}$$

$$v_F = v_{1y} = v_{3y}$$

$$v_B = v_{2y} = v_{4y}$$

اگر η را به صورت $\eta = [v_x, w]^T$ تعریف نماییم، می‌توانیم روابط زیر را نتیجه بگیریم (در آنها a, b, c پارامترهای ربات هستند که در شکل ۵.۱۶ نمایش داده شده است).

$$\begin{bmatrix} v_L \\ v_R \\ v_F \\ v_B \end{bmatrix} = \begin{bmatrix} 1 & -c \\ 1 & c \\ 0 & b - x_{ICR} \\ 0 & -a - x_{ICR} \end{bmatrix} \begin{bmatrix} v_x \\ w \end{bmatrix}$$

اگر فرض کنیم که شعاع موثر چرخ i ام، برابر با شعاع چرخ r می‌باشد، در این صورت داریم:

$$w_w = \begin{bmatrix} w_L \\ w_R \end{bmatrix} = \frac{1}{r} \begin{bmatrix} v_L \\ v_R \end{bmatrix}$$

که w_L, w_R به ترتیب سرعت زاویه ای چرخ های راست و چپ می باشد. با دو رابطه فوق می توان به رابطه زیر رسید:

$$\begin{bmatrix} v_x \\ w \end{bmatrix} = r \begin{bmatrix} \frac{w_L + w_R}{2} \\ \frac{-w_L + w_R}{2c} \end{bmatrix}$$

با در نظر گرفتن این ربات، محدودیت غیرهولونومیک را می توان به صورت $0 = v_y + x_{ICR}\dot{\theta}$ بنویسیم. بنابراین مدل سینماتیکی ربات به صورت زیر نوشته می شود:

$$\begin{aligned} \dot{q} &= S(q)\eta \\ S(q) &= \begin{bmatrix} \cos \theta & x_{ICR} \sin \theta \\ \sin \theta & -x_{ICR} \cos \theta \\ 0 & 1 \end{bmatrix} \end{aligned}$$

مدل دینامیکی

در این بخش مدل دینامیکی ربات را بیان می کنیم. اگر F_i بیانگر نیروی فعال وارد شده بر چرخ i است و N_i نشان دهنده نیروی جاذبه باشد، واضح است که نیروی فعال به طور خطی با گشتاور وارد شده بر چرخ i است یعنی؛ $F_i = \frac{\tau_i}{r}$ ، از طرفی می دانیم که نیروهای مقاوم هم چه در راستای طولی و چه در راستای عرضی وجود دارد. با تقریب می توانیم این نیروها را با اصطکاک ساده تخمین بزنیم.

$$F_f(\sigma) = \mu_c N \operatorname{sgn}(\sigma) + \mu_v \sigma$$

بنابراین نیروی های اصطکاک وارد شده بر چرخ i را می توانیم به صورت زیر بنویسیم:

$$F_{li} = \mu_{ci} m g \operatorname{sgn}(v_{yi})$$

$$F_{si} = \mu_{sci} m g \operatorname{sgn}(v_{xi})$$

که μ_{ci}, μ_{sci} ضرایب پارامترهای نیروهای اصطکاک عرضی و طولی می باشند. با استفاده از روابط لاغرانژین می توانیم به رابطه زیر برای دینامیک ربات برسیم:

$$M(q)\ddot{q} + R(q) = B(q)\tau$$

اما رابطه فوق برای مدل سازی رباتی است که هیچ محدودیتی ندارد اما با توجه به محدودیت غیرهولونومیک باید آن را بهبود بدھیم. لذا خواهیم داشت:

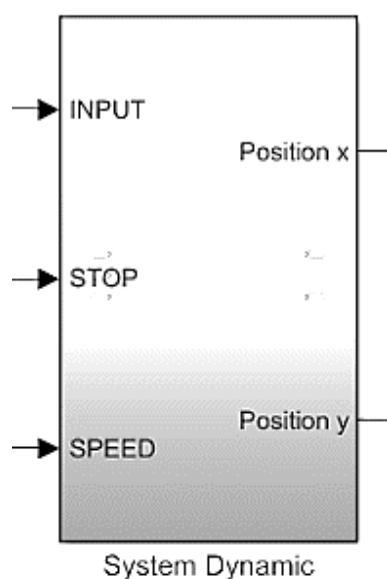
$$M(q)\ddot{q} + R(q) = B(q)\tau + A^T(q)\lambda$$

با این حال از آنجاییکه برای کنترل، بیان کردن معادله بالا بر حسب η مناسب تر می باشد، بنابراین با ضرب کردن طرفین رابطه در $(q^T S^T)$ خواهیم داشت:

$$\begin{aligned} \bar{M}(q)\dot{\eta} + \bar{C}\eta + \bar{R} &= \bar{B}\tau \\ \bar{C} = \begin{bmatrix} 0 & \dot{\theta} \\ \dot{\theta} & \dot{x}_{ICR} \end{bmatrix}, \bar{M} = \begin{bmatrix} m & 0 \\ 0 & mx^2_{ICR} + I \end{bmatrix} \\ \bar{B} = \frac{1}{r} \begin{bmatrix} 1 & 1 \\ -c & c \end{bmatrix}, \bar{R} = \begin{bmatrix} F_{rx} \\ x_{ICR}F_{ry} + M_r \end{bmatrix} \end{aligned}$$

مدل نهایی

مدل نهایی به صورت شکل ۶.۱۶ است. ورودی‌های سیستم سرعت خودرو (مقداری بین ۱۰۰ و ۱۰۰۰)، جهت آن (۱ چپ، ۲ مستقیم و ۳ راست) و توقف (۰ یا ۱) هستند. همچنین خروجی‌های اصلی سیستم نیز موقعیت‌های X, Y ربات هستند. (مقادیر منفی به معنی حرکت به سمت عقب می باشد). این مدل با پیاده سازی روابط مذکور در دو بخش قبل، ارتباط بین ورودی و خروجی‌های ربات را برقرار می کند. این مدل در نرم افزار متلب به صورت یک جعبه سیاه قرار داده شده و با اعمال انواع ورودی و خروجی، می توان مدلسازی یا طراحی کنترل کننده خودکار ربات را انجام داد. همچنین امکان کنترل دستی ربات زمینی با استفاده از همین مدل برقرار است. در ادامه به رابط گرافیکی طراحی شده در نرم افزار متلب، برای تسهیل ارتباط با این ربات به صورت دستی و خودکار معرفی می شود.



شکل ۶.۱۶: ورودی خروجی‌های UGV

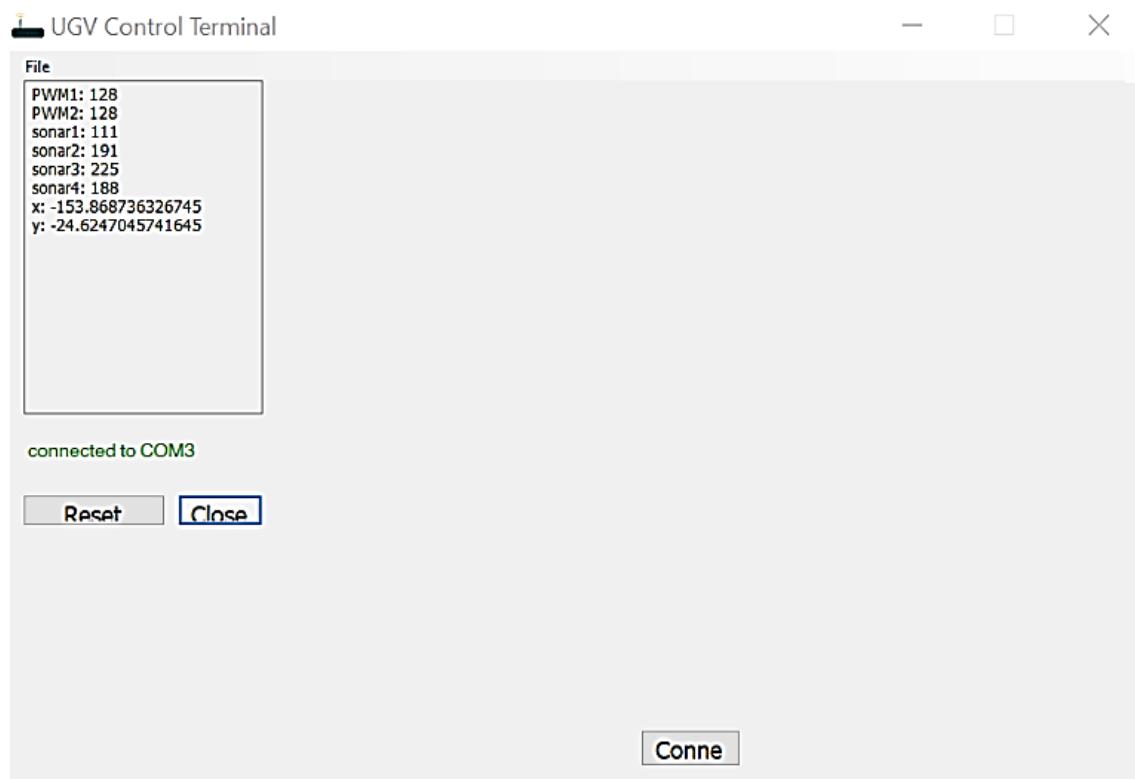
جدول ۱.۱۶: دستورات اعمالی به *UGV* برای کنترل دستی

E	Q	D	A	S	W
چپ رو به جلو	راست رو به عقب	چپ رو به عقب	راست رو به جلو	عقب	جلو

ب. ۲ ارتباط با *UGV*

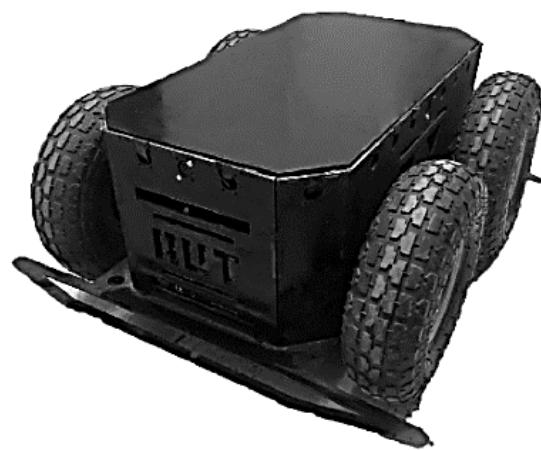
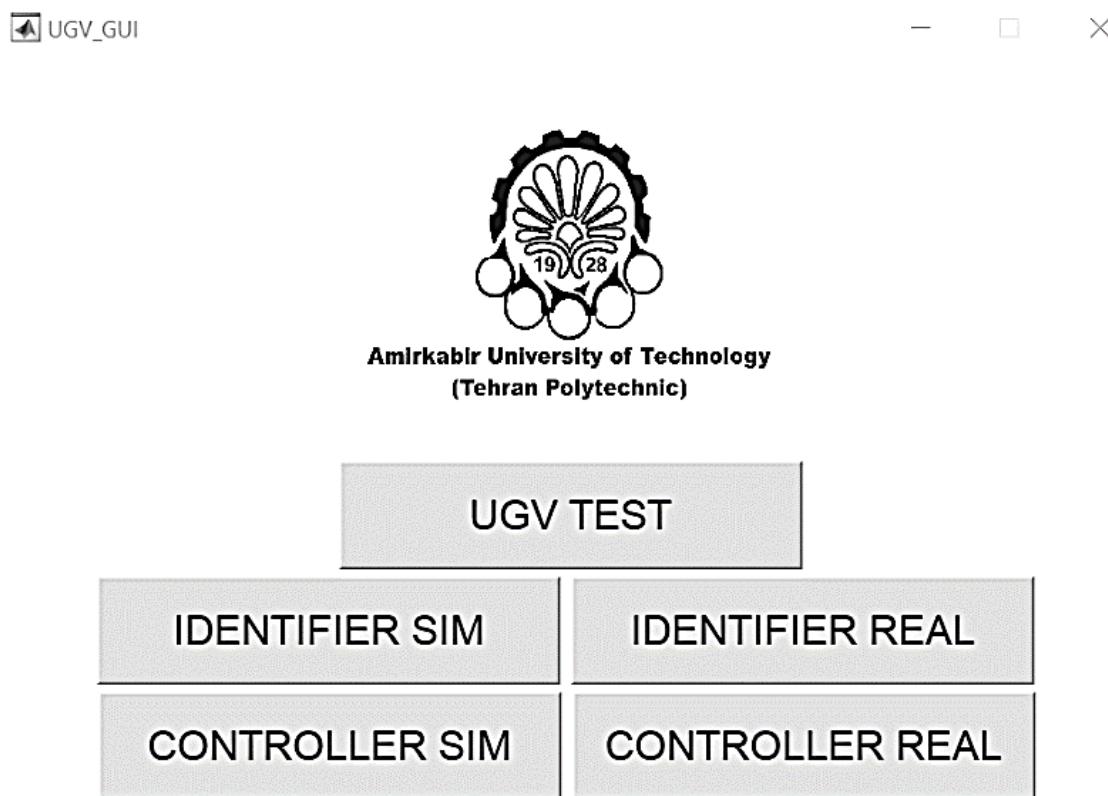
برای کار با *UGV* هاب *NRF* ارتباط با *UGV* را به کامپیوتر وصل نمایید. پیش از انجام هر کاری، پوشه *CILab* را از استاد خود گرفته باز روی کامپیوتر آزمایشگاه بردارید و در لپ تاپ خود یا آدرسی متفاوت ذخیره نمایید (در غیراین صورت محتوای برنامه‌ی خودرا از دست خواهید داد). برای باز کردن *GUI* ارتباط با *UGV* بر روی فولدر جدید *CILab* و سپس *Debug* کلیک کرده و *CMVS\UGV.exe* را باز نمایید. حال شاهدیک محیط گرافیکی خواهید بود که در صورت اتصال هاب *USB* و روشن بودن *UGV*، با فشردن دکمه *Open*، خروجی سنسورهای *UGV* را نمایش می‌دهد. با فشردن دکمه سبز رنگ بر روی ربات، امکان روشن و خاموش کردن آن وجود دارد. در صورت اضطرار، با فشردن دکمه‌های کیبورد امکان کنترل ربات به صورت دستی از طریق این صفحه مطابق جدول وجود دارد. همچنین در صورت بروز مشکل می‌توانید دکمه قرمز رنگ در جلوی ربات (فیوز ربات) را فشار دهید و ربات را متوقف کنید. برای روشن کردن مجدد ربات پس از فشردن دکمه قرمز، لازم است تا دکمه را چرخانده و به سمت بیرون هدایت کنید. فراموش نکنید که ابتدا ربات را با استفاده از دکمه سبز رنگ خاموش کرده و سپس فیوز آن را وصل کنید.

نکته: با فشردن دکمه *Reset*, مقادیر *X*, *Y*, *Z* خوانده شده توسط سنسورها برابر با صفر قرار داده می‌شوند تا محل شروع به حرکت، به عنوان مبدأ در نظر گرفته شود.



شکل ۷.۱۶: صفحه ارتباط با *UGV*

حال نرم افزار متلب را باز کرده، دکمه *Open* را فشرده و از آدرس *CILabMatlabUGVGUI.fig* را باز کنید. در این صورت با پنجره زیر مواجه خواهید شد.



شکل ۸.۱۶: محیط کاربری *UGV*

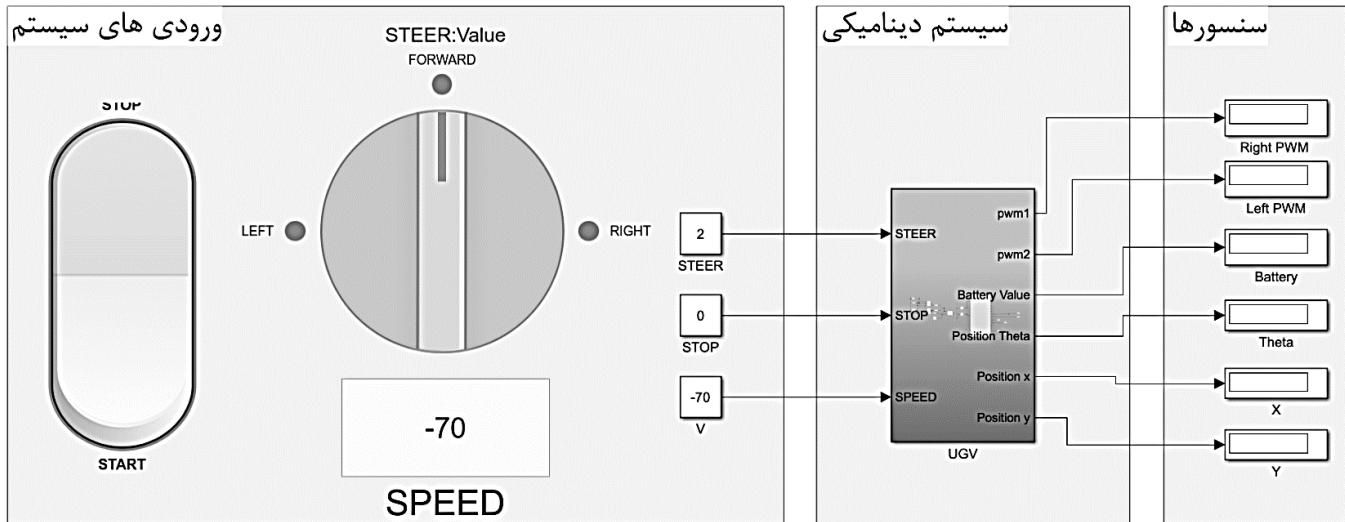
پنجره *UGVTest*

در این محیط می‌توانید با تعیین مقادیری، ورودی‌های ربات را تعریف کرده و با اجرا کردن برنامه، آن را به صورت اتوماتیک حرکت دهید.

نکته: اندازه‌ی سرعت ربات از ۰ تا ۱۰۰ است. برای حرکت هر موتور رو به جلو، از مقادیر مثبت و برای حرکت رو به عقب



Amirkabir University of Technology
(Tehran Polytechnic)



شکل ۹.۱۶: تست UGV

جدول ۲.۱۶: خروجی های سیستم

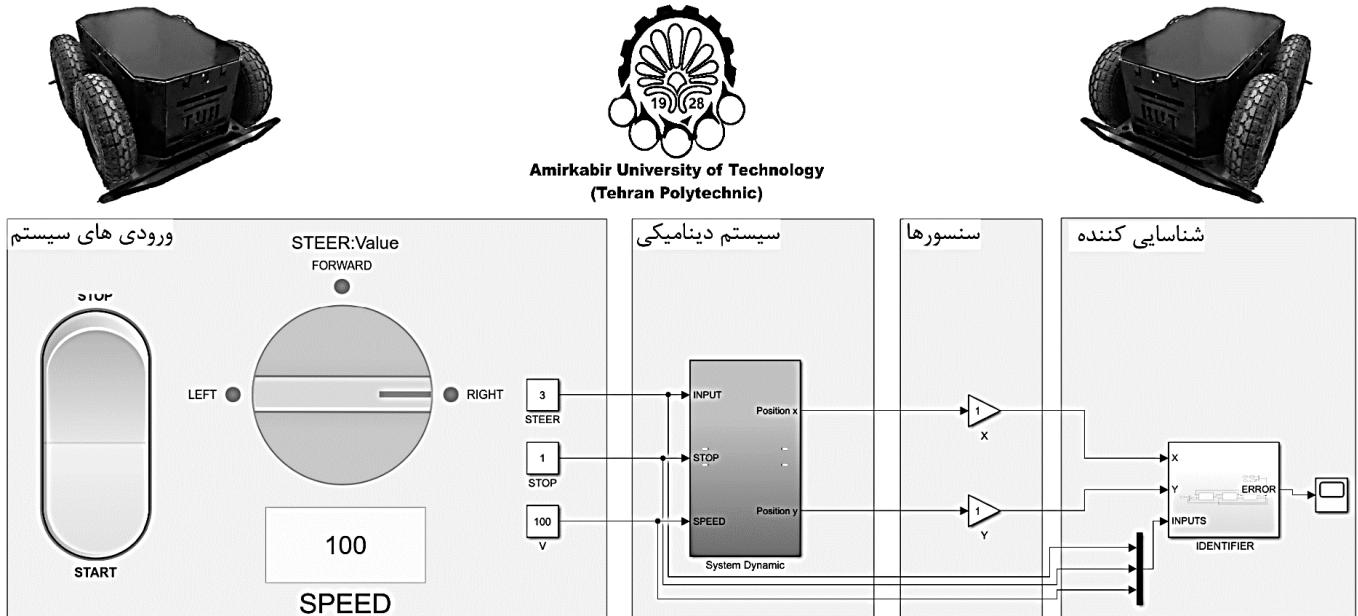
خروجی ها	X, Y	Theta	Battery	Right, Left PWMs
مقدار	موقعیت	زاویه نسبی	شارژ باتری	سرعت موتورهای راست و چپ ربات
حدوده تغییرات	ابعاد محیط	۳۶۰ تا ۰	۱۰۰ تا ۰	۲۵۵ تا ۰
واحد	متر	درجه	درصد	-

از مقادیر منفی استفاده می‌شود. همچنین جهت فرمان در سه زاویه در حین حرکت به جلو تعریف شده که شامل مقادیر ۱ برای چرخش به چپ، ۲ برای حرکت مستقیم و ۳ برای چرخش به راست است. نکته: در صورت بروز هرگونه مشکل یا برای ایجاد هرگونه تغییر، دکمه توقف را بزنید و از اجرا و متوقف کردن برنامه در حالت *START* بپرهیزید. با تعیین ورودی‌ها و فشردن دکمه *START*، در ضمن حرکت، در خروجی، مقادیر سنسورها به شرح زیرند.

نکته: به میزان درصد باتری در طول آزمایش توجه کنید. در صورت رسیدن به عدد ۰ نیاز به تعویض و شارژ باتری‌ها وجود دارد.

پنجره IDENTIFICATIONSIM

با فشردن این دکمه، صفحه‌ای مشابه زیر باز می‌شود. در این قسمت برای شما تنها امکان تغییر ورودی‌های سیستم، محتوای سیستم دینامیکی و محتوای شناساگر وجود دارد. از این بلوک فقط برای شبیه‌سازی استفاده می‌شود. توضیحات بیشتر در بخش شناسایی توضیح داده می‌شود.



شکل ۱۰.۱۶: شبیه ساز شناساگر

پنجره CONTROLSIM

با فشردن این دکمه، صفحه‌ای مشابه زیر باز می‌شود. در این قسمت برای شما تنها امکان تغییر سیگنال مرجع، محتوای سیستم دینامیکی و محتوای کنترل‌کننده‌ها وجود دارد. از این بلوک فقط برای شبیه‌سازی استفاده می‌شود. توضیحات بیشتر در دستورکار آورده شده است.

پنجره IDENTIFICATIONSIM

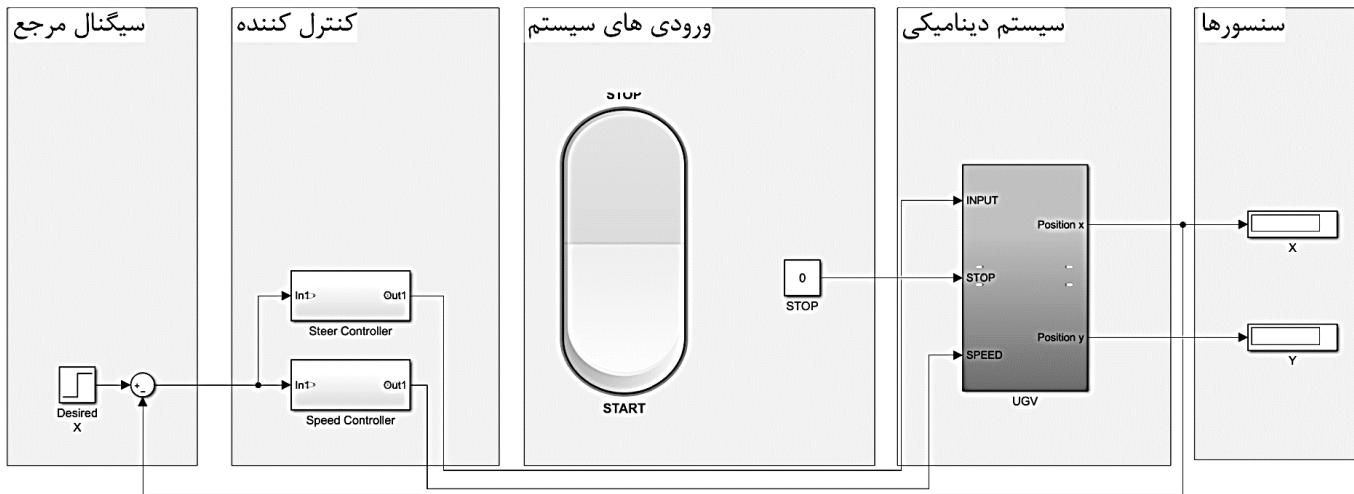
با فشردن این دکمه، صفحه‌ای مشابه زیر باز می‌شود. در این قسمت برای شما تنها امکان تغییر ورودی‌های سیستم و محتوای شناساگر وجود دارد. توضیحات بیشتر در بخش شناسایی آورده شده است.

پنجره CONTROLREAL

با فشردن این دکمه، صفحه‌ای مشابه زیر باز می‌شود. در این قسمت برای شما تنها امکان تغییر سیگنال مرجع، محتوای سیستم دینامیکی و محتوای کنترل‌کننده‌ها وجود دارد. از این بلوک فقط برای شبیه‌سازی استفاده می‌شود. توضیحات بیشتر در دستورکار آورده شده است.



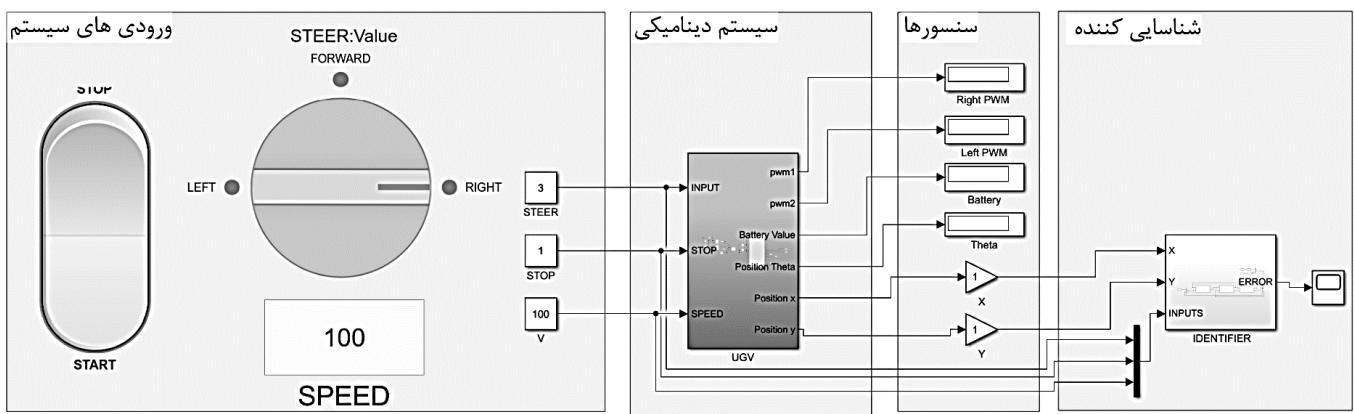
Amirkabir University of Technology
(Tehran Polytechnic)



شکل ۱۱.۱۶: شبیه ساز کنترل کننده



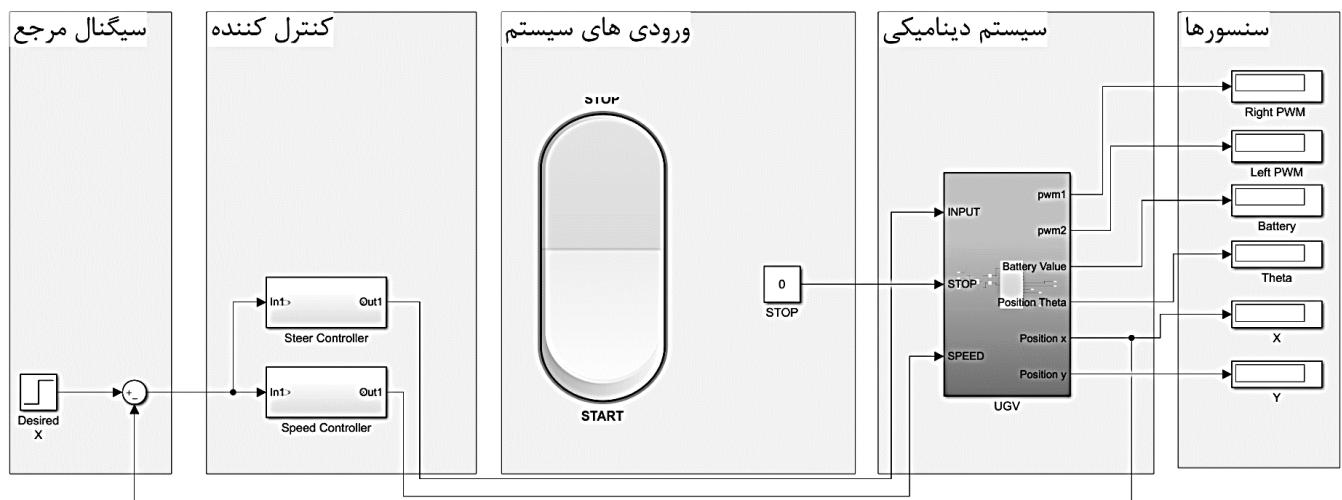
Amirkabir University of Technology
(Tehran Polytechnic)



شکل ۱۲.۱۶: شناساگر واقعی UGV



Amirkabir University of Technology
(Tehran Polytechnic)



شکل ۱۳.۱۶: کنترلر واقعی *UGV*